Feasibility of on-line speed policies in real-time systems

Bruno Gaujal, Alain Girault, Stéphan Plassart

Feasibility of on-line speed policies in real-time systems

Bruno Gaujal, Girault Alain, Stéphan Plassart

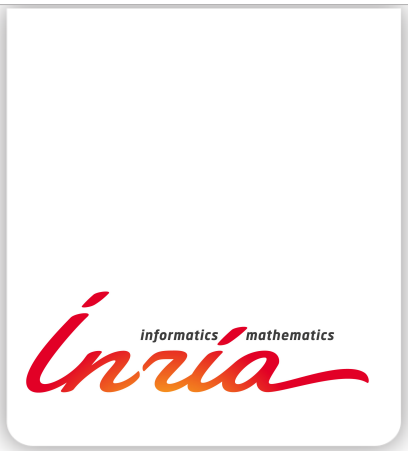
RESEARCH REPORT

N° 9301

November 2019

Project-Teams Polaris and Spades

ISSN 0249-6399 ISRN INRIA/RR--9301--FR+ENG



Feasibility of on-line speed policies in real-time systems

Bruno Gaujal, Girault Alain, Stéphan Plassart *

Project-Teams Polaris and Spades

Research Report n° 9301 — November 2019 — 38 pages

Abstract: We consider a real-time system where a single processor with variable speed executes an infinite sequence of sporadic and independent jobs. We assume that job sizes and relative deadlines are bounded by C and Δ respectively. Furthermore, $S_{\max}$ denotes the maximal speed of the processor. In such a real-time system, a speed selection policy dynamically chooses (i.e., on-line) the speed of the processor to execute the current, not yet finished, jobs. We say that an on-line speed policy is feasible if it is able to execute any sequence of jobs while meeting two constraints: the processor speed is always below $S_{\max}$ and no job misses its deadline. In this paper, we compare the feasibility region of four on-line speed selection policies in single-processor real-time systems, namely Optimal Available (OA) [1], Average Rate (AVR) [1], (BKP) [2], and a Markovian Policy based on dynamic programming (MP) [3]. We prove the following results:

- (OA) is feasible if and only if $S_{\max} \geq C(h_{\Delta-1} + 1)$, where h_n is the n-th harmonic number ($h_n = \sum_{i=1}^{n} 1/i \approx \log n$).
- (AVR) is feasible if and only if $S_{\max} \geq Ch_{\Delta}$.
- (BKP) is feasible if and only if $S_{\max} \geq eC$ (where $e = \exp(1)$).
- (MP) is feasible if and only if $S_{\max} \geq C$. This is an optimal feasibility condition because when $S_{\max} < C$ no policy can be feasible.

This reinforces the interest of (MP) that is not only optimal for energy consumption (on average) but is also optimal regarding feasibility.

Key-words: Hard Real-Time Systems, Feasibility, On-line Speed Policy, Markov Decision Process, Dynamic Voltage and Frequency Scaling.

* This work has been partially supported by the LabEx PERSYVAL-Lab

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Faisabilité des politiques en-ligne dans les systèmes temps-réel.

Résumé : Dans ce papier, on compare la faisabilité de quatre politiques de sélection de vitesses en-ligne appliquées à un processeur dans le cas des sytèmes temps-réel: Optimal Available (OA) [1], Average Rate (AVR) [1], (BKP) [2], et une politique markovienne utilisant la programmation dynamique (MP) [3]

Mots-clés : Système Temps-réel Dur, Faisabilité, Politique de Vitesse en-ligne, Processus à Decision de Markov, Adaptation en Fréquence et Ajustement Dynamique de Tension

Contents

1 Introduction **4**

2 Related work **5**

3 Presentation of the problem **5**

3.1 Hard real-time systems 5

3.2 Scheduling policy 6

3.3 On-line speed policy 6

3.4 Speed decision times 7

3.5 Feasibility problem for on-line speed policies 7

3.5.1 Feasibility Characterization 8

4 Feasibility analysis **10**

5 Feasibility of the Optimal Available speed policy (OA) **11**

5.1 Definition of (OA) [1] 11

5.2 Feasibility analysis of (OA) 12

6 Feasibility of the Average Rate speed policy (AVR) **20**

6.1 Definition of (AVR) [1] 20

6.2 Feasibility analysis 20

7 Feasibility of the Bansal, Kimbrel, Pruhs speed policy (BKP) **22**

7.1 Definition of (BKP) [2] 22

7.2 Feasibility analysis of (BKP) with $\mathcal{T} = \mathbb{N}$ 23

7.3 Feasibility analysis of (BKP) with $\mathcal{T} = \mathbb{R}$ 26

8 Feasibility of the Markov Decision Process speed policy (MP) **27**

8.1 Definition of (MP) [3] 27

8.2 Feasibility analysis of (MP) 28

9 Summary and Comparison of the four Policies **29**

10 Conclusion **31**

A Uniqueness of the solution of Eq. (38) **32**

B Concavity of the executed work by (OA) **for a given** w **34**

1 Introduction

A hard real-time system (HRTS) consists of a generally infinite sequence of independent jobs that must be executed onto some hardware platform before some strict deadline. Jobs can arrive in a periodic or sporadic manner. Such systems are found everywhere today: in energy production, in transport (automotive, avionics, ...), in embedded systems, to name only a few application domains. Each job is characterized by its arrival time, its size *i.e.*, the amount of work to complete the job, and its strict deadline, either defined absolutely or relatively to the arrival time. We consider the particular case of unconstrained HRTS executed on a single core processor with variable processor speed. An HRST is therefore characterized by a tuple $(C, \Delta, S_{\max})$, where C is the maximal size of the jobs, Δ is their maximal deadline, and $S_{\max}$ is the maximal speed of the processor. The inter-arrival times between the jobs are unconstrained (*i.e.*, neither periodic or sporadic).

Changing the speed of the processor can help to reduce the energy consumption of the processor, which is essential in many embedded systems. In fact, this is the reason why modern processors are equipped with Dynamic Voltage and Frequency Scaling (DVFS) capabilities. Several speed selection policies have been proposed to save energy by modifying the speed of the processor on-line. The main idea behind all on-line speed policies is to lower the speed when the current load is low, in order to save energy and, when the load is high, to increase the speed to execute all jobs before their deadlines.

In this article, the main goal is to analyze the feasibility of existing on-line speed policies. A policy is feasible if and only if each job is executed before its absolute deadline. Without loss of generality, we assume that the time scale is discrete and that a new job arrives at each time step. In contrast, the processor speed can change at any time.

The first on-line speed policy that comes to mind involves, at each time step, executing entirely the current job within one time step. Obviously this policy is feasible, because all the jobs finish before their deadline. Moreover, the maximal processor speed used under this policy is not larger than C. Therefore, this policy is feasible if $S_{\max} \geq C$. This is optimal in terms of feasibility because no policy can be feasible when $S_{\max} < C$: indeed, if $S_{\max} < C$, then a job of size C with deadline 1 will miss its deadline. In contrast, regarding the energy consumption, this policy consumes more than any other policy because it does not take advantage of job deadlines (assuming that the energy is an increasing convex function, which is usually the case). For these reasons, we analyze in this article the feasibility of known policies that lower the energy consumption.

We investigate the four following on-line speed policies. To the best of our knowledge, these are the four such existing speed policies. The first two ones are (AVR) and (OA), both from [1], which both try to optimize the energy consumption of a real-time system. The third one is (BKP) from Bansal et al. [2], the goal of which is to improve the competitive ratio of (OA). The fourth one is a Markov Decision Process policy called (MP) in the rest of the paper, which optimizes the expected energy consumption when statistical information on the arrival, WCET, and deadline of the jobs are available [3].

In their original respective paper, the authors of (AVR), (OA), and (BKP) all make the unrealistic assumption that $S_{\max}$ is unbounded, *i.e.*, $S_{\max} = +\infty$. Under this assumption, feasibility is not as problematic: all jobs can be executed before their deadline as long as the current selected speed is large enough. However, under the more realistic assumption of a bounded $S_{\max}$, one needs to compute the feasibility region in the parameter space of $(C, \Delta, S_{\max})$. Our goal in this paper is therefore to determine, for the classical policies (AVR), (OA), (BKP), and for (MP),

the maximal speed $S_{\max}$ as a function of C and Δ, that ensures feasibility, and to compare the four policies in this respect.

The paper is organized as follows. We survey the related work in Section 2. Then we present the job model used in Section 3 and formulate the feasibility analysis problem in Section 4. In the subsequent sections we analyze each on-line speed policy and we prove, for each of them, what is the smallest value of $S_{\max}$ that ensures feasibility (Sections 5 to 8). Finally, we compare the four on-line speed policies based on these values $S_{\max}$ in Section 9 before concluding in Section 10.

2 Related work

The work that is most closely related to our is [4], which investigates the feasibility of (AVR) and (OA) (this latter speed policy being called (OPT) in their paper). The system model is a single-core processor that must execute an infinite sequence real-time jobs, specified by an *arrival curve* (as in the Real-Time Calculus [5]). Arrival curves generalize both the periodic task model and the sporadic task model with minimal inter-arrival time. An important assumption in [4] is that all the jobs have the same WCET C and the same relative deadline Δ. The main result is that, both for (OA) and (AVR), the feasibility condition is $S_{max} \geq \frac{\alpha^u(\Delta)}{\Delta}$, where α^u is the upper arrival curve of the sequence of jobs, meaning that $\alpha^u(D)$ is an upper bound on the work that can arrive during any time interval of length D. Actually, the same feasibility condition applies to (BKP) and (MP), although these speed policies are not studied in [4]. In contrast to this result, we do not constrain the jobs to have the same WCET nor the same deadline. Therefore the analysis becomes completely different as well as the feasibility conditions which are now different for each policy.

To the best of our knowledge, all the other results on feasibility analysis of on-line speed policies found in the literature target system models either with a fixed inter-arrival time between the jobs (*i.e.*, periodic tasks) or with a bounded inter-arrival time (*i.e.*, sporadic tasks). Papers in this category are plentyful, let us just cite [6] in the periodic case and [7] in the sporadic case. In constrast, we make no assumption on the inter-arrival times between jobs.

3 Presentation of the problem

3.1 Hard real-time systems

We consider an HRTS that executes an infinite sequence of sporadic and independent jobs $\{J_i\}_{i\in\mathbb{N}}$ on a single-core processor with varying frequency. Each job J_i is defined as a tuple (r_i, c_i, d_i) where $r_i \in \mathbb{N}$ is the release time (or arrival time), $c_i \in \mathbb{N}$ is the size (also called workload), *i.e.*, the amount of work to complete the job, and $d_i \in \mathbb{N}$ is the absolute deadline of job J_i, satisfying $d_i > r_i$. The jobs are ordered by their release times. Their relative deadlines are $D_i := d_i - r_i$, *i.e.*, the amount of time given to the processor to execute the job. The jobs are sporadic, meaning that their arrival times do not follow any particular pattern. This is the most general model of jobs.

We further assume that all jobs have a bounded relative deadline: there exists Δ such that

$$\forall i, D_i = d_i - r_i \leq \Delta \tag{1}$$

where Δ is the maximal relative deadline. Several jobs may arrive simultaneously but in any

case the cumulated size is assumed to be bounded by C. In other words:

$$\forall t, \sum_{i \,|\, r_i = t} c_i \leq C. \tag{2}$$

Finally, we denote by $\mathcal{J}_{C,\Delta}$ the set of all possible sequences of jobs that satisfy the two assumptions stated in Eqs. (1) and (2).

Definition 1 (Set of all possible sequences of jobs: $\mathcal{J}_{C,\Delta}$)**.**

$$\mathcal{J}_{C,\Delta} := \left\{ J = \{J_i = (r_i, c_i, d_i)\}_{i \in \mathbb{N}} \middle| \forall t, \sum_{i \,|\, r_i = t} c_i \leq C \ \wedge \ \forall i, d_i - r_i \leq \Delta \right\}. \tag{3}$$

Minimality of the assumptions. Let us anticipate a bit on what follows and comment about the relevance of the two assumptions stated by Eq. (1) and (2). We claim that these are the minimal assumptions under which feasibility of a speed policy can be asserted.

First, in most practical cases, the set of jobs comes from a finite set of tasks (infinite sequences of jobs with the same features). In this case, relative deadlines and sizes are always bounded. Besides, if the set of jobs is finite, then everything is bounded.

Consider now the most general case, *i.e.*, with an infinite set of sporadic jobs. If the relative deadlines are not bounded, then the set of pending jobs at some arbitrary time t cannot be bounded and the time needed to compute the current speed for all on-line policies is also unbounded, so that feasibility cannot be asserted in finite time.

Once the condition that all jobs have a bounded deadline is stated, the assumption on the arriving work (2) must also be made. Indeed, if a set of jobs arrives at time t, all with deadlines bounded by Δ, and brings an unbounded amount of work into the system, then no speed policy with a given maximal speed will be able to execute this work before time $t + \Delta$.

3.2 Scheduling policy

At any time $t \in \mathbb{R}$, several jobs may be active (*i.e.*, released and not yet finished). In this case we must choose which job to execute first on the single-core processor. This ordering is known as a *schedule* and the policy for making this choice is known as the *scheduling policy.*

Definition 2 (Schedule feasibility)**.** *A schedule is* feasible *over an infinite sequence of jobs* $J = \{(r_i, c_i, d_i)\}_{i \in \mathbb{N}} \in \mathcal{J}_{C,\Delta}$ *if and only if each job* (r_i, c_i, d_i) *is executed between its release time and its absolute deadline,* i.e.*, between* r_i *and* d_i*.*

It has been shown that the Earliest Deadline First (EDF) scheduling policy is optimal for feasibility [8], meaning that if a sequence J is feasible for some scheduling policy, then it is also feasible under EDF. Therefore, in the following, we will always assume that the processor uses EDF to schedule its active jobs.

3.3 On-line speed policy

In most modern processors, the speed (or frequency) can be adjusted dynamically. This can be achieved with DVFS, a technology available on most of today's processors. In this paper, we

use the term "speed" instead of "frequency" to reflect the fact that the processor performs some work quantity per time unit: More precisely, when operating at speed s, the amount of work performed by the processor during one time unit is equal to s.

We focus on on-line speed policies, the goal of which is to choose, at each time t, the speed at which the processor should run, based on the current information (we assume that no look-ahead is available).

Given a sequence of jobs $J = \{(r_i, c_i, d_i)\}_{i\in\mathbb{N}}$ and the speeds $s(t)$ used at all time $t \in \mathbb{R}$, we define the history of the system up to time t.

Definition 3 (History)**.** *The history of the system up to time t is:*

$$\mathcal{H}_t = \{J_i, r_i \leq t\} \cup \{s(u), u \leq t\}. \tag{4}$$

All the release times, job sizes, and deadlines are integer numbers. Therefore, the sequence of jobs $\{J_i, r_i \leq t\}$ only changes at integer time instants. This is not the case for the processor speeds $\{s(u), u \leq t\}$, which can change at any time instant. We will detail this in Section 3.4.

Definition 4 (On-line speed policy)**.** *An on-line* speed policy π *is a function that assigns, at time t with the history $\mathcal{H}_t$, a speed s to the processor:*

$$\pi(\mathcal{H}_t, t) = s. \tag{5}$$

In the following, we will often use $\pi(t)$ to simplify the notation, but one should keep in mind the fact that, in full generality the speed selected at time t may depend on t, the jobs that arrived before t, and the speeds selected before t.

Since the maximal speed of the processor is $S_{\max}$, any speed policy π must satisfy the following constraint:

$$\forall t, \forall J,\ 0 \leq \pi(\mathcal{H}_t, t) \leq S_{\max}. \tag{6}$$

3.4 Speed decision times

We define as *speed decision times* the times at which the processor speed can change. These times do not necessarily coincide with the job arrival times. For instance, processor speeds may change several times between two potential job arrivals. In the rest of this article, we study the two different cases:

- The processor speed changes can only occur when a job arrives: $t \in \mathbb{N}$.
- The processor speed changes can occur at any time: $t \in \mathbb{R}$.

In the following, we denote by $\mathcal{T}$ the set of speed decision times. As discussed above, the two possible cases are studied in this article: $\mathcal{T} = \mathbb{N}$ and $\mathcal{T} = \mathbb{R}$. For (OA), (AVR), and (MP), we will show that the cases $\mathcal{T} = \mathbb{N}$ and $\mathcal{T} = \mathbb{R}$ yield the same feasibility conditions. For (BKP), the two cases are slightly different.

3.5 Feasibility problem for on-line speed policies

The goal of our article is to determine the condition for which feasibility is satisfied for several speed policies.

Definition 5 (Speed policy's feasibility)**.** *An* on-line speed policy π *is feasible over an infinite sequence of jobs* $J = \{(r_i, c_i, d_i)\}_{i\in\mathbb{N}}$ *if and only if when the processor runs at speed* $\pi(t)$ *for all* t *and uses EDF, each job* (r_i, c_i, d_i) *is executed before its absolute deadline:*

$$\pi \text{ is feasible} \Longleftrightarrow \left(\sup_{J\in\mathcal{J}_{C,\Delta}} \sup_{t\in\mathcal{T}} \pi(t) \leq S_{max} \right) \wedge \text{no missed deadline.} \tag{7}$$

In Eq. (7), the second term "no missed deadline" is not very explicit. For this reason we redefine it by using the remaining work function, which is presented next. In the rest of the paper we use the following notation: x_+ is the positive part of x: $x_+ := \max(x, 0)$.

Definition 6 (Remaining work function)**.** *The* remaining work function *under* π *at time* t *is the function* $w_t^\pi(\cdot)$*, such that, at any future time* $u \geq t$*, the* remaining work $w_t^\pi(u)$ *is the amount of work that has arrived by time* t *whose deadline is before* u*, minus the amount of work already executed at time* t*. It satisfies a Lindley's equation by induction:*

$$\begin{cases} w_0^\pi(u) = 0 & \forall u \geq 0 \\ w_t^\pi(u) = \left(w_k^\pi(u) - \int_k^t \pi(v)dv \right)_+ + A(t,u) & \forall k \in \mathbb{N} \text{ with } k < t \leq k+1 \\ & \text{and } \forall u \geq t > 0 \end{cases} \tag{8}$$

where $A(t, u)$ *is the amount of work corresponding to the jobs arriving at time* t *whose deadline is smaller or equal to* u*.*

Two remarks are in order:

Remark 1. *The arrival function* $A(t, u)$ *is equal to* 0 *if* $t \notin \mathbb{N}$*, because the release times of all jobs are in* $\mathbb{N}$*.*

Remark 2. *Since the maximal job relative deadline is* Δ*,* $w_t^\pi(t + \Delta)$ *is the total amount of remaining work at time* t*. In other words,* w_t^π *increases up to time* t_Δ *and stays constant after that time* $t + \Delta$*:* $\forall u \geq t + \Delta$*,* $w_t^\pi(u) = w_t^\pi(\Delta + t)$*. Moreover, for any online policy* π*,* $\int_k^{k+1} \pi(v)dv \leq w_k^\pi(k + \Delta)$ *because, at time* k*, the processor can only execute work present in the system at time* k*. By straightforward induction, this implies:*

$$w_t^\pi(t + \Delta) = \sum_{r_i \leq t} c_i - \int_0^t \pi(v)dv \tag{9}$$

and when no deadlines are missed, then:

$$w_t^\pi(t + \Delta) \leq C\Delta. \tag{10}$$

3.5.1 Feasibility Characterization

Using Def. (8) of the remaining work function, one can make the definition of feasibility given in Def. (5) more explicit. For this purpose, we state Prop. 1 that links the remaining work function and the policy. This proposition introduces a new condition of feasibility.

Proposition 1.

$$\pi \text{ is feasible} \Longleftrightarrow \left(\sup_{J\in\mathcal{J}_{C,\Delta}} \sup_{t\in\mathcal{T}} \pi(t) \leq S_{max} \right) \wedge \left(\forall J \in \mathcal{J}_{C,\Delta}, \forall t \in \mathcal{T},\ w_t^\pi(t) = 0 \right). \tag{11}$$

The first condition says that the speed selected by π at t must always be smaller than $S_{\max}$, while the second condition says that at any t, all the work whose deadline is before t has already been executed. Although this may seem trivial, let us write an explicit proof of this equivalence.

Proof. We rely on the definition of feasibility given in Def. 5. There are two parts in this definition, and to prove the proposition, we will begin to show that:

$$\text{no missed deadline} \iff \forall J \in \mathcal{J}_{C,\Delta}, \forall t \in \mathcal{T},\ w_t^\pi(t) = 0. \tag{12}$$

The proof of Eq. (12) is divided in two parts, each of them proves one implication.

1. No missed deadline $\Longrightarrow \forall t, 0 = w_t^\pi(t)$:

   By contraposition, let us show that $0 < w_t^\pi(t) \Longrightarrow$ missed deadline. If $0 < w_t^\pi(t)$, then it means that some work whose deadline is before t has not been executed by time t, so at least one job has missed its deadline before time t.

2. $\forall t, 0 = w_t^\pi(t) \Longrightarrow$ no missed deadline:

   If $0 = w_t^\pi(t)$, then at each time t, all the work whose deadline was before t has been executed. Thanks to EDF, we know that all the jobs whose deadline is exactly at time t have been executed before t. This is true for all t, so it is also true for all the jobs.

The condition involving $S_{\max}$ is the same as in the original definition. □ □

The following proposition establishes a necessary condition of the feasibility for any on-line speed policy π.

Proposition 2. *For decision times $\mathcal{T} = \mathbb{N}$ and $\mathcal{T} = \mathbb{R}$ and for any policy π, a necessary condition of feasibility is:*

$$S_{\max} \geq C. \tag{13}$$

Proof. Let π be any feasible on-line speed policy and let J be the sequence of jobs made of the single job $J_0 = (0, C, 1)$. By Def. 6, $w_1^\pi(1) = (C - \int_0^1 \pi(v)dv)_+$. The second part of the feasibility condition of π says that at time 1, $w_1^\pi(1)$ must be equal to 0. This implies $\int_0^1 \pi(v)dv \geq C$. Since $\int_0^1 \pi(v)dv \leq \max_{0\leq t\leq 1} \pi(t)$, we therefore have, $\max_{0\leq t\leq 1} \pi(t) \geq C$. Then, the first part of the feasibility condition implies that $S_{\max} \geq \max_{0\leq t\leq 1} \pi(t)$. Putting both parts together yields $S_{\max} \geq C$. □ □

Proposition 3. *In the case of integer decision times ($\mathcal{T} = \mathbb{N}$), the condition $\forall t \in \mathbb{R}$, $w_t^\pi(t) = 0$ can be re-written as $\forall k \in \mathbb{N}$, $\pi(k) \geq w_k^\pi(k+1)$.*

Proof. The proof simply follows the definitions. When the speed is constant in the interval $[k, k+1)$,

$$w_{k+1}^\pi(k+1) \quad = \quad (w_k^\pi(k+1) - \pi(k))_+ + A(k+1, k+1),$$

with $A(k+1, k+1) = 0$ because jobs arriving at time $k+1$ have a deadline at least $k+2$. Hence:

$$w_{k+1}^\pi(k+1) \quad = \quad (w_k^\pi(k+1) - \pi(k))_+$$

It follows that $w_{k+1}^\pi(k+1) = 0$ if and only if $(w_k^\pi(k+1) - \pi(k))_+ = 0$. By definition of the max, this is equivalent to $\pi(k) \geq w_k^\pi(k+1)$. □ □

Table 3.5.1 summarizes all the notations used in the paper.

J_i	Job number i
$r_i, c_i, d_i \in \mathbb{N}$	Release time, size, and absolute deadline of job i
$D_i = d_i - r_i$	Relative deadline of job i
Δ	Bound on all relative deadlines
C	Bound on the work amount arriving at any time t
$\mathcal{J}_{C,\Delta}$	Set of all sequences of jobs with bounds C and Δ
$S_{\max}$	Maximal speed of the processor
$\pi(t)$	Speed used by the processor at time t
$\mathcal{T}$	Time instants when the processor can change its speed (here, $\mathcal{T} = \mathbb{N}$ or $\mathcal{T} = \mathbb{R}$)
$w_t^\pi(u), t \leq u$	Remaining work under speed policy π: at time t, it is the amount of pending work to be executed before time u
$A(t, v), t \leq v$	Work arriving at t with deadline smaller than v
h_n	The n-th harmonic number: $h_n = \sum_{i=1}^{n} 1/i = \log(n) + \gamma + o(1/n)$
$\mathcal{F}_\pi$	Set of all $(C, \Delta, S_{\max})$ such that policy π is feasible over any sequence of jobs in $\mathcal{J}_{C,\Delta}$ with a maximal speed $S_{\max}$
$u(t, t_1, t_2)$	Amount of work arrived after t_1 and before t

Table 1: Notations used throughout the paper.

4 Feasibility analysis

The goal of this article is to study the feasibility of the four different on-line speed policies (OA), (AVR), (BKP), and (MP). For each policy, we formally establish a necessary and sufficient feasibility condition on $S_{\max}$. In each case, the proof follows the same route. We first check that if $S_{\max} = \infty$ then the policiy is feasible. This part of the proof is already provided in the papers introducing the policies, but we briefly sketch them when the argument is trivial. Then, still assuming that $S_{\max} = \infty$, we compute the maximal speed $\overline{\pi}$ used by the online policy under a worst case sequence of jobs in $\mathcal{J}_{C,\Delta}$. Therefore, a necessary and sufficient condition of feasibility is $S_{\max} \geq \overline{\pi}$. We construct such a worst case sequence for each policy. While these worst case sequences will look similar (at least the first three), the analysis relies on very different techniques:

- The proof for (OA) policy uses a construction (Lindley's equation, with a backward construction) that comes from queueing theory (Section 5).
- The proof for (AVR) is based on the explicit construction of a worst case, which consists of a maximal number of jobs that have the same deadline (Section 6).
- The proof for (BKP) exploits arithmetic considerations (Section 7).
- The proof for (MP) is based on a dynamic programming analysis (Section 8).

At any time t, the (OA) and (MP) policies both compute the processor speed based on the work remaining at t, while the (AVR) and (BKP) policies do not. This is in part why the proofs are so diverse. As a final note before starting with the proofs, the case of (OA) is by far the more interesting. In spite of the apparent simplicity of (OA), the proof uses several backward inductions as well as properties of generalized differential equations (with non-differentiable functions).

5 Feasibility of the Optimal Available speed policy (OA)

5.1 Definition of (OA) [1]

Definition 7 (Optimal Available (OA))**.** *At each time $t \in \mathcal{T}$, the job that has the earliest deadline is executed at speed:*

$$\pi^{(\mathrm{OA})}(t) = \max_{v>t} \left(\frac{w_t^{(\mathrm{OA})}(v)}{v-t} \right) \tag{14}$$

where $w_t^{(\mathrm{OA})}(.)$ is the remaining work defined in Def. 6.

To illustrate (OA), let us consider the following set of jobs with $\mathcal{T} \in \mathbb{N}$, which is composed of 3 jobs and belongs to $\mathcal{J}_{4,5}$:

- $J_1 = (r_1 = 0, c_1 = 1, d_1 = 4)$ hence $D_1 = 4$,
- $J_2 = (r_2 = 3, c_2 = 4, d_2 = 6)$ hence $D_2 = 3$,
- $J_3 = (r_3 = 3, c_3 = 1, d_3 = 8)$ hence $D_3 = 5$,

Let us compute the (OA) speed at time 3. According to Eq. (14), it is equal to:

$$\pi^{(\mathrm{OA})}(3) = \max_{v>3} \left(\frac{w_3^{(\mathrm{OA})}(v)}{v-3} \right)$$

At each of the three instants 0, 1, and 2, only the job J_1 is present, so the speed computed by Eq. (14) is equal to:

$$\pi^{(\mathrm{OA})}(0) = \pi^{(\mathrm{OA})}(1) = \pi^{(\mathrm{OA})}(2) = \frac{c_1}{D_1} = \frac{1}{4}.$$

Therefore, at time 3, we have:

$$\begin{aligned}
\pi^{(\mathrm{OA})}(3) &= \max \left\{ \frac{w_3^{(\mathrm{OA})}(4)}{d_1-3}, \frac{w_3^{(\mathrm{OA})}(6)}{d_2-3}, \frac{w_3^{(\mathrm{OA})}(8)}{d_3-3} \right\} \\
&= \max \left\{ \frac{c_1 - \pi^{(\mathrm{OA})}(0) - \pi^{(\mathrm{OA})}(1) - \pi^{(\mathrm{OA})}(2)}{d_1-3}, \right. \\
&\qquad \frac{c_1 + c_2 - \pi^{(\mathrm{OA})}(0) - \pi^{(\mathrm{OA})}(1) - \pi^{(\mathrm{OA})}(2)}{d_2-3}, \\
&\qquad \left. \frac{c_1 + c_2 + c_3 - \pi^{(\mathrm{OA})}(0) - \pi^{(\mathrm{OA})}(1) - \pi^{(\mathrm{OA})}(2)}{d_3-3} \right\} \\
&= \max \left\{ \frac{1}{4}, \frac{17}{12}, \frac{21}{20} \right\}
\end{aligned}$$

In conclusion, we have $\pi^{(\mathrm{OA})}(3) = \frac{17}{12}$.

5.2 Feasibility analysis of (OA)

In this section, we will determine the smallest maximal processor speed $S_{\max}$ that guarantees the feasibility of (OA). Theorem 1 gives a necessary and sufficient feasibility condition for (OA).

Theorem 1. (OA) *is feasible* $\iff S_{\max} \geq C(h_{\Delta-1}+1)$*, where* h_n *is the* n*-th harmonic number:* $h_n = \sum_{i=1}^{n} 1/i$.

Proof. We distinguish the cases where the speed decision times are integer and real numbers.

♦ **The speed decision times are integer numbers:** $\mathcal{T} = \mathbb{N}$**.**
In the integer case, Eq. (14) becomes:

$$\pi^{(\mathrm{OA})}(t) = \max_{v\in\mathbb{N},\, v>t} \left(\frac{w_t^{(\mathrm{OA})}(v)}{v-t} \right). \tag{15}$$

By taking $v = t+1$, Eq. (15) implies that $\pi^{(\mathrm{OA})}(t) \geq w_t^{(\mathrm{OA})}(t+1)$. Therefore, the feasibility Equation (11) can be written as a condition on $S_{\max}$ only:

$$\text{(OA) is feasible} \iff \forall t,\, S_{\max} \geq \pi^{(\mathrm{OA})}(t).$$

The rest of the proof is structured as follows. (i) We will first derive a bound on $\pi^{(\mathrm{OA})}(t)$ (steps 1, 2, and 3). (ii) Then we will construct an explicit worst-case scenario that reaches this bound asymptotically.

Let us first compute an upper bound on the remaining work $w_t^{(\mathrm{OA})}(v)$, for any $t \in \mathbb{N}$ and any integer $v > t$. This will be done in several steps. To simplify notations, in the following, we denote $\pi^{(\mathrm{OA})} = \pi$ and $w^{(\mathrm{OA})} = w$, since the only speed policy considered hare is (OA) and no confusion is possible.

We can focus on times $v \leq t+\Delta$ because the remaining work after time $t+\Delta$ remains the same (see Remark 2). Now, $w_t(v)$ only depends on three things:

- the remaining work function at time $v-\Delta$: $w_{v-\Delta}(\cdot)$,
- the work that arrives between times $v-\Delta+1$ and t,
- and the speeds used at times $v-\Delta$ to $t-1$.

The definition of $w_t(v)$ yields:

$$\begin{aligned}
w_t(v) &= \big(w_{t-1}(v) - \pi(t-1)\big)_+ + A(t,v) && (16)\\
w_{t-1}(v) &= \big(w_{t-2}(v) - \pi(t-2)\big)_+ + A(t-1,v) && (17)\\
&\vdots \\
w_{v-\Delta+1}(v) &= \big(w_{v-\Delta}(v) - \pi(v-\Delta)\big)_+ + A(v-\Delta+1,v). && (18)
\end{aligned}$$

This first shows that the function w_t increases when π decreases.

Step 1: The first step amounts to showing that $w_t(v)$ becomes larger if the sizes of all the jobs whose absolute deadline is larger than v are set to 0, while keeping the rest unchanged.

This fact is easy to check: In Eqs (16)-(18), the only terms that depend on those jobs are the speeds. Under (OA), the speeds are increasing with the remaining work. Therefore, by removing these jobs, all the speeds are decreased (or remain the same) and $w_t(v)$ is increased.

Step 2: The second step amounts to checking that, if the remaining work function $w_{v-\Delta}(\cdot)$ is replaced by the function $w^*_{v-\Delta}(\cdot)$ such that (i) $w^*_{v-\Delta}(i) = 0$ for $i = v-\Delta+1, \ldots, v-1$ and $w^*_{v-\Delta}(v) = w_{v-\Delta}(v)$, and such that (ii) all jobs arriving at times $v-\Delta < i \leq t$ have their deadline set at v, then this change increases the remaining work at time v. This construction is illustrated in Fig. 1 where the $w^*_{v-\Delta}(\cdot)$ function is depicted by the black curve.

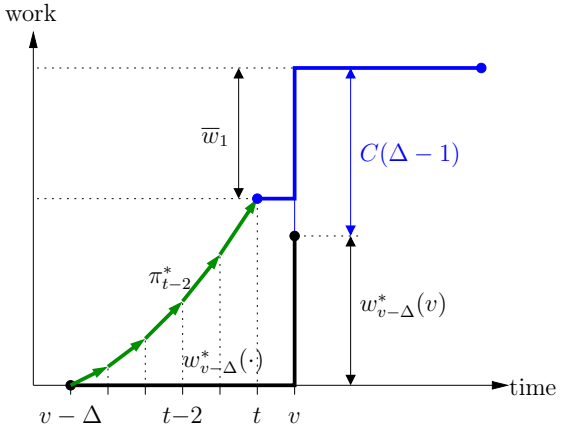


Figure 1: Construction of $w^*_t(v)$ for $v = t+1$ and $\Delta = 6$. The bold black curve is the lower bound on the remaining work $w^*_{v-\Delta}(\cdot)$. The bold blue curve is the final upper bound $\overline{w}_1$ on the remaining work. The bold green arrows represent the work executed by the processor at each time slot i at speed $\pi^*(i)$.

We will show this by induction (putting a star on all values computed with the new work function $w^*_{v-\Delta}(\cdot)$):

- Initial step $i = 0$: $w^*_{v-\Delta}(v) \geq w_{v-\Delta}(v)$ by definition of w^*.
- Induction assumption at step i:

$$w^*_{v-\Delta+i}(v) \geq w_{v-\Delta+i}(v) \tag{19}$$

- Let us prove the induction property at step $i+1$, i.e., that $w^*_{v-\Delta+i+1}(v) \geq w_{v-\Delta+i+1}(v)$. Let $h := w^*_{v-\Delta+i}(v) - w_{v-\Delta+i}(v)$. We first have:

$$\pi^*(v-\Delta+i) = \frac{w^*_{v-\Delta+i}(v)}{\Delta-i}$$

because, at any time $r < v$, we have $w^*_{v-\Delta+i}(r) = 0$ by construction.

For the original system, $\pi(v-\Delta+i) \geq \frac{w_{v-\Delta+i}(v)}{\Delta-i}$ because the maximum could be reached

for some $r < v$. This yields:

$$\begin{aligned} w_{v-\Delta+i}(v) - \pi(v-\Delta+i) &\leq w_{v-\Delta+i}(v) - \frac{w_{v-\Delta+i}}{\Delta - i} \\ &= w^*_{v-\Delta+i}(v) - h - \frac{w^*_{v-\Delta+i}(v) - h}{\Delta - i} \\ &= w^*_{v-\Delta+i}(v) - \frac{w^*_{v-\Delta+i}(v)}{\Delta - i} - \left(h - \frac{h}{\Delta - i}\right) \\ &\leq w^*_{v-\Delta+i}(v) - \frac{w^*_{v-\Delta+i}(v)}{\Delta - i} \\ &= w^*_{v-\Delta+i}(v) - \pi^*(v-\Delta+i). \end{aligned} \tag{20}$$

Furthermore, for each i, $w_{v-\Delta+i}(v)$ is the total amount of work present in the original system at time $v-\Delta+i$, because we have discarded all jobs with deadline larger than v in Step 1. This implies $\pi(v-\Delta+i) \leq w_{v-\Delta+i}(v)$, hence:

$$\big(w_{v-\Delta+i}(v) - \pi(v-\Delta+i)\big)_+ = w_{v-\Delta+i}(v) - \pi(v-\Delta+i). \tag{21}$$

Putting Eqs. (20) and (21) together, since for all $k \leq t$, $A^*(k,v) = A(k,v)$, we get:

$$\begin{aligned} w^*_{v-\Delta+i+1}(v) &= \big(w^*_{v-\Delta+i}(v) - \pi^*(v-\Delta+i)\big)_+ + A^*(v-\Delta+i+1,v) \\ &\geq \big(w_{v-\Delta+i}(v) - \pi(v-\Delta+i)\big) + A(v-\Delta+i+1,v) \\ &= \big(w_{v-\Delta+i}(v) - \pi(v-\Delta+i)\big)_+ + A(v-\Delta+i+1,v) \\ &= w_{v-\Delta+i+1}(v), \end{aligned}$$

which is the property we wanted to prove at step $i+1$. This finishes Step 2.

Step 3: In the star system (work function $w^*_{v-\Delta}(\cdot)$), the speeds used by (OA) at times $v-\Delta$ to $t-1$ are respectively:

$$\begin{aligned} \pi^*(v-\Delta) &= \frac{w^*_{v-\Delta}(v)}{\Delta} \\ \pi^*(v-\Delta+1) &= \frac{w^*_{v-\Delta}(v)}{\Delta} + \frac{A(v-\Delta+1,v)}{\Delta-1} \\ \pi^*(v-\Delta+2) &= \frac{w^*_{v-\Delta}(v)}{\Delta} + \frac{A(v-\Delta+1,v)}{\Delta-1} + \frac{A(v-\Delta+2,v)}{\Delta-2} \\ &\vdots \\ \pi^*(t-1) &= \frac{w^*_{v-\Delta}(v)}{\Delta} + \frac{A(v-\Delta+1,v)}{\Delta-1} + \frac{A(v-\Delta+2,v)}{\Delta-2} + \cdots + \frac{A(t-1,v)}{v-t+1}. \end{aligned}$$

We introduce the variable $u = v - t$ and compute the sum of all speeds:

$$\sum_{i=v-\Delta}^{t-1} \pi^*(i) = \frac{(\Delta-u)w^*_{v-\Delta}(v)}{\Delta} + \frac{(\Delta-u-1)A(v-\Delta+1,v)}{\Delta-1} + \cdots + \frac{A(t-1,v)}{u+1}. \tag{22}$$

We then compute the sum of Eqs. (16) to (18), in the case of the star system. Note that the speeds $\pi^*(i)$ never become larger than the work $w^*(i)$, so the max operator is never "active" and

can be removed:

$$w_t^*(v) = w_{v-\Delta}^*(v) - \sum_{i=v-\Delta}^{t-1} \pi^*(i) + \sum_{i=v-\Delta+1}^{t} A(i,v). \tag{23}$$

By replacing in Eq. (23) the sum of the speeds (Eq. (22)), we obtain the remaining work at $v = t + u$:

$$w_t^*(v) \quad = \quad \frac{uw_{v-\Delta}^*(v)}{\Delta} + \frac{uA(v-\Delta+1,v)}{\Delta-1} + \cdots + \frac{uA(t-1,v)}{u+1} + A(t,v). \tag{24}$$

Since $w_{v-\Delta}^*(v) \leq C\Delta$ (see Eq. (9)) and $A(k,v) \leq C$ for all $k \leq t$, we obtain an upper bound on $w_t^*(t+u)$:

$$w_t^*(t+u) \quad \leq \quad \frac{uC\Delta}{\Delta} + \frac{uC}{\Delta-1} + \cdots + \frac{uC}{u+1} + C. \tag{25}$$

This finishes Step 3 and provides a bound on $w_t(t+u)$, for all t, because $w_t(t+u) \leq w_t^*(t+u)$.

To find an upper bound on $\pi^{(\mathrm{OA})}(t)$, we divide by u:

$$\frac{w_t(t+u)}{u} \leq \frac{C\Delta}{\Delta} + \frac{C}{\Delta-1} + \cdots + \frac{C}{u+1} + \frac{C}{u}.$$

The bound on the right hand side of Eq. (5.2) is maximal when $u = 1$. We therefore get an upper bound on $\pi^{(\mathrm{OA})}(t)$, denoted $\overline{w}_1$:

$$\pi^{(\mathrm{OA})}(t) \leq \underbrace{C + \frac{C}{\Delta-1} + \cdots + \frac{C}{2} + C}_{\overline{w}_1}. \tag{26}$$

The star remaining function w^* (as displayed in Figure 1) is not reachable under (OA). However, one can construct a remaining work function that is asymptotically arbitrarily close to it. This construction is illustrated in Figure 2. First, jobs of size C and relative deadline Δ arrive at each slot during n time slots. When n grows to infinity, the speed selected by (OA) approaches C and the remaining work approaches the black staircase displayed in Figure 2 (see Lemma 1 below). Then, jobs of size C and absolute deadline $\Delta + n$ arrive at all time slots from $n+1$ to $n+\Delta-1$ (job arrivals are represented alternatively in blue and red in Figure 2). In that case, we will show that $w_{n+\Delta-1}^{(\mathrm{OA})}(n+1)$ will approach $\overline{w}_1$ as n goes to infinity.

Lemma 1. *If the sequence of jobs is such that at each time n a job arrives with size C and relative deadline Δ, then:*

- *The speeds $\pi^{(\mathrm{OA})}(n)$ increase and converge towards C when n goes to infinity;*
- *The remaining work function converges towards the function $w_n(\cdot)$ such that $\forall i \leq \Delta$, $w_n(n+i) = iC$.*

Proof. We show by induction on n that $w_n(n+\Delta-1) \leq C(\Delta-1)$ and that $\pi(n) = w_n(n+\Delta)/\Delta$.

- Initial step $n = 0$: a single job has arrived, with size C and deadline Δ. Therefore, $w_0(\Delta-1) = 0 \leq C(\Delta-1)$ and $\pi(0) = w_0(\Delta)/\Delta = C/\Delta$.

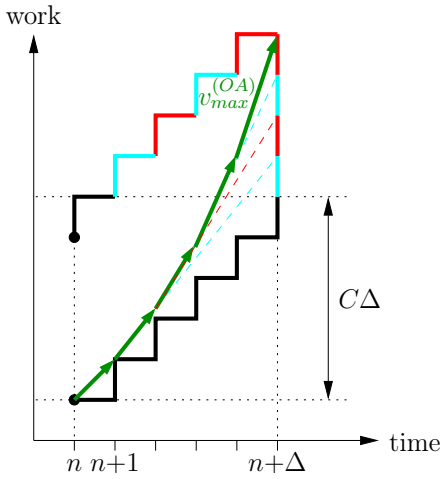


Figure 2: Asymptotic worst case state, for $C = 1$ and $\Delta = 5$. The staircase black curve represents the remaining work function reached asymptotically while the coloured parts (blue and red segments represent one job at each time slot with identical WCET C and identical absolute deadline) lead to the maximal $w^{(\mathrm{OA})}_{n+\Delta-1}(n+\Delta)$. The green arrows represent the quantities of work executed by the processor under (OA).

- Induction assumption at step n:

$$w_n(n+\Delta-1) \le C(\Delta-1) \quad \wedge \quad \pi(n) = w_n(n+\Delta)/\Delta. \tag{27}$$

- Let us prove the induction property at step $n+1$. We have:

$$\begin{aligned} w_{n+1}(n+\Delta) &= (w_n(n+\Delta) - \pi(n))_+ + A(n+1, n+\Delta) \\ &= w_n(n+\Delta) - \frac{w_n(n+\Delta)}{\Delta} + 0 \\ &= w_n(n+\Delta)\frac{\Delta-1}{\Delta}. \end{aligned}$$

Since $w_n(n+\Delta)$ is always smaller than $C\Delta$ (see Eq. (9)), it follows that:

$$w_{n+1}(n+\Delta) \le C(\Delta-1). \tag{28}$$

Let us now compute the speed π at time $n+1$. Since the speed at time n is $\pi(n) = \max_v \frac{w_n(n+v)}{v} = \frac{w_n(n+\Delta)}{\Delta}$, which is not reached for $v = 1$, then $\pi(n+1) = \max(\pi(n), w_{n+1}(n+\Delta+1)/\Delta)$. Replacing $\pi(n)$ by its value from the induction hypothesis yields:

$$\begin{aligned} \pi(n+1) &= \max\left(\frac{w_n(n+\Delta)}{\Delta}, \frac{w_{n+1}(n+\Delta+1)}{\Delta}\right) \\ &= \frac{1}{\Delta}\max\big(w_n(n+\Delta), w_{n+1}(n+\Delta+1)\big). \end{aligned} \tag{29}$$

Since the job that arrives at time $n+\Delta+1$ is of size C, the second term of the max is:

$$w_{n+1}(n+\Delta+1) = w_{n+1}(n+\Delta) + C = w_n(n+\Delta)\frac{\Delta-1}{\Delta} + C.$$

Using the induction assumption. It follows that:

$$w_{n+1}(n+\Delta+1) = w_n(n+\Delta) + \left(C - \frac{w_n(n+\Delta)}{\Delta}\right).$$

We again use the fact that $w_n(n+\Delta) \leq C\Delta$ (see Eq. (9)) to conclude that the second term of the max is always larger than the first term, giving:

$$\pi(n+1) = \frac{w_{n+1}(n+\Delta+1)}{\Delta}. \tag{30}$$

This ends the induction and shows as a byproduct that $\pi(n+1) \geq \pi(n)$.

Now, since $\pi(n)$ is increasing, it converges to some value $L \leq \infty$. Since for all n, $0 \leq w_n(n+\Delta) \leq c\Delta$, and since $w_n(n+\Delta) = \sum_{i=0}^{n} C - \sum_{i=0}^{n-1} \pi(i)$ is equivalent to $n(C-L)$ when n grows, then $L = C$.

As for the second part of the Lemma, it follows from inspecting Eq. (29). The fact that $\pi(n+1) - \pi(n)$ goes to 0, implies that $w_{n+1}(n+\Delta)$ goes to $C(\Delta-1)$. This implies that $w_{n+1}(n+1+i)$ goes to Ci, for all $0 \leq i \leq \Delta$. This concludes the proof of Lemma 1. □ □

In the following we use the following notation: $x_n \approx y_n$ if $|x_n - y_n| \leq \varepsilon$, for some $\varepsilon > 0$ arbitrarily small.

Let us now resume the proof of Theorem 1. Consider the following job sequence: First n jobs arrive, with release times $1, 2, \ldots n$, size C and *relative* deadline Δ, the next jobs arrive at times $n+1, n+2, \ldots n+\Delta-1$ with size C and *absolute* deadline $n+\Delta$. We assume that n is large enough so that using Lemma 1, $w_n(n+i) \approx Ci$ for all $i \leq \Delta$ and $\pi^{(\mathrm{OA})}(n) \approx C$ (see Figure 2).

By construction of the job sequence,

- at time $n+1$, $\pi^{(\mathrm{OA})}(n+1) \approx \dfrac{C\Delta}{\Delta-1}$;
- more generally, for $1 \leq k < \Delta$, we have on the one hand:

$$\begin{aligned}
&(\Delta-k)\pi^{(\mathrm{OA})}(n+k) \approx (\Delta+k-1)C - \sum_{j=1}^{k-1} \pi^{(\mathrm{OA})}(n+j) \\
\iff\ &(\Delta-k-1)\pi^{(\mathrm{OA})}(n+k) \approx (\Delta+k-1)C - \sum_{j=1}^{k-1} \pi^{(\mathrm{OA})}(n+j) - \pi^{(\mathrm{OA})}(n+k) \\
\iff\ &(\Delta-k-1)\pi^{(\mathrm{OA})}(n+k) \approx (\Delta+k-1)C - \sum_{j=1}^{k} \pi^{(\mathrm{OA})}(n+j)
\end{aligned} \tag{31}$$

and on the other hand:

$$(\Delta-k-1)\pi^{(\mathrm{OA})}(n+k+1) \approx (\Delta+k)C - \sum_{j=1}^{k} \pi^{(\mathrm{OA})}(n+j). \tag{32}$$

By subtracting Eq. (31) to Eq. (32), we obtain:

$$\begin{aligned}
&(\Delta-k-1)\left(\pi^{(\mathrm{OA})}(n+k+1) - \pi^{(\mathrm{OA})}(n+k)\right) \approx C \\
\iff &\pi^{(\mathrm{OA})}(n+k+1) \approx \pi^{(\mathrm{OA})}(n+k) + \frac{C}{\Delta-k-1}.
\end{aligned} \tag{33}$$

By applying iteratively Eq. (33) from $n+k+1$ down to $n+1$, we obtain for all $k \geq 1$:

$$\begin{aligned}\pi^{(\mathrm{OA})}(n+k+1) &\approx \pi^{(\mathrm{OA})}(n+1) + C\sum_{j=2}^{k+1}\frac{1}{\Delta-j}\\ &\approx \pi^{(\mathrm{OA})}(n+1) + C(h_{\Delta-2} - h_{\Delta-k-2})\end{aligned}$$

where h_n is the n-th harmonic number: $h_n = \sum_{i=1}^{n} 1/i$ and $h_0 = 0$. Therefore $\pi^{(\mathrm{OA})}(n+\Delta-1)$ has the following asymptotic value:

$$C\left(\frac{\Delta}{\Delta-1} + h_{\Delta-2}\right) = C\left(1 + h_{\Delta-1}\right) = \overline{w}_1,$$

by using $\pi^{(\mathrm{OA})}(n+1) \approx C\frac{\Delta}{\Delta-1}$.

To conclude, the (OA) policy may use a speed arbitrarily close to its upper bound, $\overline{w}_1$. Therefore, it is feasible if and only if

$$S_{\max} \geq \overline{w}_1 = C\left(1 + h_{\Delta-1}\right). \tag{34}$$

This concludes the proof of Theorem 1 in the case $\mathcal{T} = \mathbb{N}$. □

♦ **The speed decision times are real numbers: $\mathcal{T} = \mathbb{R}$.**
We will prove that, when (OA) is given the opportunity to change the speed at any time $t \in \mathbb{R}$, the speed chosen at any real time t is the same as the speed chosen at the previous integer instant.

Let us denote by $w_k^{\mathbb{N}}$ the remaining work under integer decision times, and $w_k^{\mathbb{R}}$ the remaining work under real decision times. We will prove by induction on k that for any integer k, $w_k^{\mathbb{R}} = w_k^{\mathbb{N}}$.

For the sake of simplicity, let us denote as π instead of $\pi^{\mathbb{R}}$ the (OA) speed function when the speeds can change at any real instant.

For all $k \in \mathbb{N}$, for all t such that $k < t < k+1$ and all $v \geq t$, we recall the formula giving the remaining work, when the current time is t (see Eq. (8)):

$$w_t^{\mathbb{R}}(v) = \left(w_k^{\mathbb{R}}(v) - \int_k^t \pi(x)\mathrm{d}x\right)_+ + A(t,v) = \left(w_k^{\mathbb{R}}(v) - \int_k^t \pi(x)\mathrm{d}x\right)_+ \tag{35}$$

and according to Def. 7 of the (OA) policy, at each time $t \in \mathbb{R}$ we have:

$$\pi(t) = \max_{v>t}\left(\frac{w_t^{\mathbb{R}}(v)}{v-t}\right). \tag{36}$$

Combining Eqs. (35) and (36) yields:

$$\pi(t) = \max_{v>t}\frac{\left(w_k^{\mathbb{R}}(v) - \int_k^t \pi(x)\mathrm{d}x\right)_+}{v-t}. \tag{37}$$

The $(\cdot)_+$ operator can be removed in Eq. (37) because, for $v = k+\Delta$, $w_k^{\mathbb{R}}(k+\Delta) \geq \int_k^t \pi(x)dx$ (see Remark 2). It follows that:

$$\pi(t) = \max_{v>t}\frac{\left(w_k^{\mathbb{R}}(v) - \int_k^t \pi(x)\mathrm{d}x\right)}{v-t}. \tag{38}$$

Let us now prove by induction on k that $\forall k \in \mathbb{N}$, $w_k^{\mathbb{R}} = w_k^{\mathbb{N}}$.

- Initial step $k = 0$: only the first job J_1 may have arrived at time 0. Therefore, for all $v \geq 0$, $w_0^{\mathbb{R}}(v) = w_0^{\mathbb{N}}(v) = c_1$ if $r_1 = 0, d_1 \leq v$ and $w_0^{\mathbb{R}}(v) = w_0^{\mathbb{N}}(v) = 0$ otherwise.
- Induction assumption at step k:
$$w_k^{\mathbb{R}} = w_k^{\mathbb{N}}. \tag{39}$$
- Now consider $t \in \mathbb{R}$ such that $k < t < k + 1$. We first prove that $\pi(t) = \pi(k)$.
We define m as
$$m := \operatorname*{argmax}_v \frac{w_k^{\mathbb{R}}(v)}{v - k}. \tag{40}$$
This means that
$$\pi(k) = \frac{w_k^{\mathbb{R}}(m)}{m - k}. \tag{41}$$
Now let us check whether a constant speed on $[k, k + 1)$ — *i.e.*, $\pi(t) = \pi(k), \forall t \in [k, k + 1)$ — can be a solution of Eq. (38), the integral equation defining π. With a constant speed, the numerator in Eq. (38) becomes:
$$w_t^{\mathbb{R}}(v) = w_k^{\mathbb{R}}(v) - \pi(k)(t - k). \tag{42}$$
By using the value of $\pi(k)$ from Eq. (41), we obtain:
$$\frac{w_t^{\mathbb{R}}(v)}{v - t} = \frac{w_k^{\mathbb{R}}(v)(m - k) - w_k^{\mathbb{R}}(m)(t - k)}{(v - t)(m - k)}. \tag{43}$$
First, the particular case where $v = m$ leads to:
$$\frac{w_t^{\mathbb{R}}(m)}{m - t} = \frac{w_k^{\mathbb{R}}(m)(m - k) - w_k^{\mathbb{R}}(m)(t - k)}{(m - t)(m - k)} = \frac{w_k^{\mathbb{R}}(m)}{m - k} = \pi(k). \tag{44}$$
Second, we show that this particular case is also the maximal value for $w_t^{\mathbb{R}}(v)/(v - t)$. Eq. (41) implies that: $\frac{w_k^{\mathbb{R}}(v)}{v-k} \leq \frac{w_k^{\mathbb{R}}(m)}{m-k}$. Together with Eq. (43), this yields:
$$\begin{aligned} \forall v \in \mathbb{R}, \quad \frac{w_t^{\mathbb{R}}(v)}{v - t} &= \frac{w_k^{\mathbb{R}}(v)(m - k) - w_k^{\mathbb{R}}(m)(t - k)}{(v - t)(m - k)} \\ &\leq \frac{w_k^{\mathbb{R}}(m)(v - k) - w_k^{\mathbb{R}}(m)(t - k)}{(v - t)(m - k)} \\ &= \frac{w_k^{\mathbb{R}}(m)(v - t)}{(v - t)(m - k)} \\ &= \pi(k). \end{aligned} \tag{45}$$
By Appendix (A), the solution of Eq. (38) is unique. Therefore the solution of this equation is:
$$\forall t \in [k, k + 1) \quad \pi(t) = \pi(k). \tag{46}$$
Since the speed is constant between two integer time steps, and since, by the induction assumption (39), $w_k^{\mathbb{R}} = w_k^{\mathbb{N}}$, we thus have $w_{k+1}^{\mathbb{R}} = w_{k+1}^{\mathbb{N}}$. This concludes the induction proof.

This induction proof also shows that the speed decision are the same for integer and real decision time. This implies that the behaviour of (OA) with real decision times is the same as the behaviour of (OA) with integer decision times. Therefore the feasibility condition is the same. This concludes the proof of Theorem 1. □ □

6 Feasibility of the Average Rate speed policy (AVR)

6.1 Definition of (AVR) [1]

(AVR) is defined in [1] as follows:

Definition 8 (AVerage Rate (AVR))**.** *At each time $t \in \mathcal{T}$, the job that has the earliest deadline is executed at speed:*

$$\pi^{(\mathrm{AVR})}(t) = \sum_{i \in \mathcal{A}(t)} \frac{c_i}{d_i - r_i} \tag{47}$$

where $\mathcal{A}(t)$ is the set of active *jobs at time t,* i.e.*, jobs $J_i = (r_i, c_i, d_i)$ such that $r_i \leq t < d_i$.*

Notice that the processor speed $\pi^{(\mathrm{AVR})}(t)$ is independent of the previous speeds used by the processor. In contrast, (OA) chooses at time t a speed that, through $w^{(\mathrm{OA})}$, depends on the previous speeds used by the processor.

Let us apply (AVR) policy on the example displayed in Section (5), where we consider the same 3 jobs:

- $J_1 = (r_1 = 0, c_1 = 1, d_1 = 4)$
- $J_2 = (r_2 = 3, c_2 = 4, d_2 = 6)$
- $J_3 = (r_3 = 3, c_3 = 1, d_3 = 8)$

The three jobs are active at time 3, thus using Eq. (47) yields:

$$\pi^{(\mathrm{AVR})}(3) = \frac{c_1}{d_1 - r_1} + \frac{c_2}{d_2 - r_2} + \frac{c_3}{d_3 - r_3} = \frac{1}{4-0} + \frac{4}{6-3} + \frac{1}{8-3}$$

Therefore $\pi^{(\mathrm{AVR})}(3) = \frac{107}{60}$.

We note that the speed chosen at time 3 by (AVR) is greater than the one chosen by (OA). However, in the next section, we will show that the maximal speed required by (AVR) for feasibility is smaller than the maximal speed required by (OA) and determined in Section 5.

6.2 Feasibility analysis

Theorem 2 establishes the condition on $S_{\max}$ that insures the feasibility of (AVR).

Theorem 2. (AVR) *is feasible* $\iff S_{\max} \geq Ch_{\Delta}$.

Proof. We distinguish the cases where the decision times are integer and real numbers.

♦ **The decision times are integer numbers: $\mathcal{T} = \mathbb{N}$.**
According to Prop. 1, the (AVR) feasibility proof is split in two different parts.

▶ The first part consists in showing that all jobs are executed before their deadlines, *i.e.*, $\pi^{(\mathrm{AVR})}(t) \geq w_t^{(\mathrm{AVR})}(t+1)$.

Let us focus on one job $J_i = (r_i, c_i, d_i)$. Under (AVR), one can consider that the processor dedicates a fraction of its computing power to execute a quantity of work equal to $\frac{c_i}{D_i}$ per time unit from r_i to $r_i + D_i - 1$, for job J_i only. So at time $r_i + D_i$, the job J_i is totally executed by the processor, hence before its deadline. Since this reasoning is valid for all jobs, all jobs are executed before their deadline under (AVR) as long as $S_{\max}$ is large enough.

Therefore, the feasibility equation (11) can be simplified and written as a condition on $S_{\max}$:

$$\text{(AVR) is feasible} \iff \forall t \in \mathcal{T},\ S_{\max} \geq \pi^{(\mathrm{AVR})}(t). \tag{48}$$

▶ Let us now compute the minimal value of $S_{\max}$ such that (AVR) is feasible, by building a worst-case scenario:

- By definition of (AVR), there is no influence of the work already executed on the value of the current speed. We therefore focus on the currently active jobs.
- $\pi^{(\mathrm{AVR})}(t)$ increases with the size of each job, so we consider jobs of maximal size, namely C.
- $\pi^{(\mathrm{AVR})}(t)$ increases with the number of active jobs, so our worst-case scenario involves the maximal possible number of active jobs, namely Δ (because only one job of size C can arrive at each time step, with a deadline not larger than Δ).
- $\pi^{(\mathrm{AVR})}(t)$ increases when the deadline of the jobs are small, so we consider jobs with the smallest possible deadline, namely $t+1$.

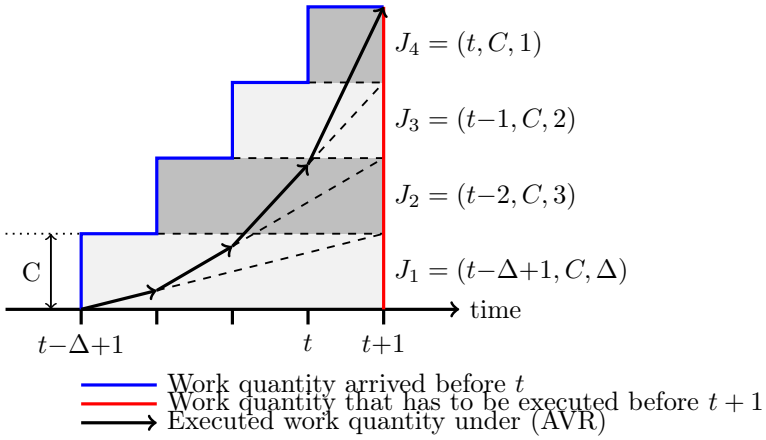


Figure 3: Worst-case scenario for (AVR) when $\Delta = 4$.

In this worst-case scenario (illustrated in Fig. 3 when $\Delta = 4$), the speed $\pi^{(\mathrm{AVR})}(t)$ is maximal at time t and is the sum of the average speed of each active job J_i executed separately:

$$\pi^{(\mathrm{AVR})}(t) = \sum_{i=1}^{\Delta} \frac{C}{i} = C h_\Delta.$$

It follows that the feasibility condition for (AVR) is:

$$Ch_\Delta \leq S_{\max}.$$

This worst-case scenario allows us to determine the maximal processor speed $S_{\max}$ under which the (AVR) policy can schedule any sequence of jobs without missing a deadline. If we suppose that $S_{\max} \leq Ch_\Delta$, then there exists a job configuration on which (AVR) is not feasible, as shown in Fig 3. Therefore Theorem 2 is proved.

♦ **The speed decision times are real numbers: $\mathcal{T} = \mathbb{R}$.**
By definition, $\pi^{(\mathrm{AVR})}(t)$ only depends on the set of active jobs, satisfying $r_i \leq t < d_i$. Since r_i and d_i are integer numbers, the set of active jobs is the same for t and for $\lfloor t \rfloor$. As for the previous policy, allowing real decision times for (AVR) does not change the chosen speeds. We thus have the same feasibility condition for the integer and real decision times, which is $S_{\max} \geq Ch_\Delta$.
□ □

7 Feasibility of the Bansal, Kimbrel, Pruhs speed policy (BKP)

7.1 Definition of (BKP) [2]

Definition 9 (Contributing work)**.** *For any t, t_1, and t_2 in $\mathbb{R}$ such that $t_1 \leq t \leq t_2$, $u(t, t_1, t_2)$ is the amount of work arrived after t_1 and before t, the deadline of which is less than t_2.*

According to Def. 9, any job $J_i = (r_i, c_i, d_i)$ contributing to $u(t, t_1, t_2)$ must satisfy $t_1 \leq r_i \leq t$ and $d_i \leq t_2$.

Definition 10 (Bansal, Kimbrel, Pruhs policy (BKP))**.** *At each time t, the job that has the earliest deadline is executed at speed:*

$$\pi^{(\mathrm{BKP})}(t) = \max_{t_2 > t} \left\{ \frac{u(t, et - (e-1)t_2, t_2)}{t_2 - t} \right\}. \tag{49}$$

Remark 3. (BKP) *was designed to improve the competitive ratio of* (OA)*, from α^α for* (OA) *to $2(\frac{\alpha e}{\alpha - 1})^\alpha$ for* (BKP)*, when the power dissipated by the processor at speed s is s^α [2].*

Let us apply the policy (BKP) on the simple example displayed in Sections 5 and 6. We recall the 3 jobs:

- $J_1 = (r_1 = 0, c_1 = 1, d_1 = 4)$,
- $J_2 = (r_2 = 3, c_2 = 4, d_2 = 6)$,
- $J_3 = (r_3 = 3, c_3 = 1, d_3 = 8)$.

For computing the max in Eq. (49) at time 3, let us examine four possible cases:

1. $t_2 > d_3$: In that case, the 3 jobs are present at time u, hence:

$$\frac{u(3, 3e - (e-1)t_2, t_2)}{t_2 - 3} \leq \frac{c_1 + c_2 + c_3}{d_3 - 3} = \frac{6}{5}.$$

2. $d_2 < t_2 < d_3$: Because of the deadlines, in the best case, only 2 jobs J_1 and J_2 are present at time u, hence:
$$\frac{u(3, 3e-(e-1)t_2, t_2)}{t_2-3} \leq \frac{c_1+c_2}{d_2-3} = \frac{5}{3}.$$

3. $t_2 = d_2$: The 2 jobs J_1 and J_2 are present at time u, hence:
$$\frac{u(3, 3e-(e-1)t_2, t_2)}{t_2-3} = \frac{c_1+c_2}{d_2-3} = \frac{5}{3}.$$

4. $t_2 < d_2$: Only job J_1 can be present at time u, hence:
$$\frac{u(3, 3e-(e-1)t_2, t_2)}{t_2-3} \leq \frac{c_1}{d_1-3} = 1.$$

As a consequence, we obtain:
$$\pi^{(\mathrm{BKP})}(3) = \frac{c_1+c_2}{d_2-3} = \frac{5}{3}.$$

The following table summarizes the numerical values computed by the three speed policies (OA), (AVR), and (BKP) at time 3 and for the chosen example with three jobs.

(OA)	(AVR)	(BKP)
17/12	107/60	5/3

We therefore have the following inequality:
$$\pi^{(\mathrm{OA})}(3) \leq \pi^{(\mathrm{BKP})}(3) \leq \pi^{(\mathrm{AVR})}(3).$$

In the following, we will show that even if the behavior of (BKP) looks like a compromise between (OA) and (AVR), the feasibility condition of (BKP) is much better than both.

7.2 Feasibility analysis of (BKP) with $\mathcal{T} = \mathbb{N}$

Theorem 3. (BKP) *is feasible with* $\mathcal{T} = \mathbb{N} \iff S_{\max} \geq \frac{3}{2}(e-1)C$.

Proof. From Theorem 5 in [2], (BKP) completes all the jobs by their deadlines. As a consequence, $\forall t \in \mathbb{R}$ (and hence in $\mathbb{N}$), we have $\pi^{(\mathrm{BKP})}(t) \geq w_t^{(\mathrm{BKP})}(t+1)$. Therefore, the feasibility equation, Eq. (11), can be simplified and rewritten as a condition only on $S_{\max}$:
$$\text{(BKP) is feasible with } \mathcal{T} = \mathbb{N} \iff \forall t \in \mathbb{N},\ S_{\max} \geq \pi^{(\mathrm{BKP})}(t). \tag{50}$$

In order to prove Condition (50), we will find an upper and a lower bound for the maximal speed of (BKP). To find an upper bound on $S_{\max}^{(\mathrm{BKP})}$, we have to determine an upper bound on $u(t, t_1, t_2)$. Let $t \in \mathbb{N}$. We split the analysis in two cases:

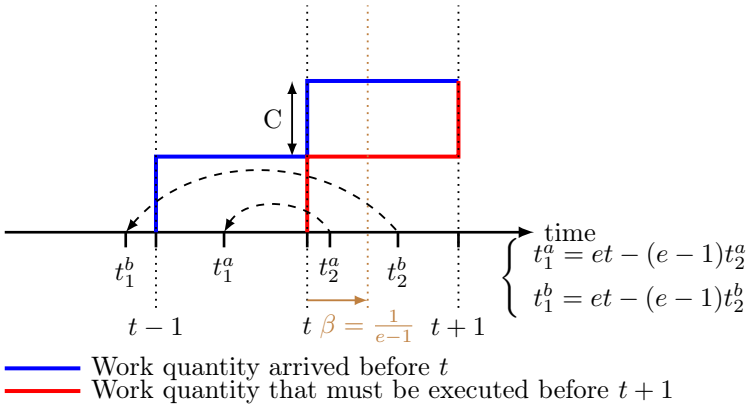


Figure 4: (BKP) with integer speed decision times ($t \in \mathbb{N}$) and Case 1 ($t_2 < t+1$). Let $t_2^i \in (t, t+1]$ with $i \in \{a, b\}$ to illustrate the two sub-cases. Before $t+\beta$ (sub-case $i = a$), no jobs are taken into account in the speed computation, so $S_{\max}^{(\mathrm{BKP})} = 0$. After this threshold (sub-case $i = b$), $S_{\max}^{(\mathrm{BKP})}$ can be non null because we take potentially into account the job arriving at $t-1$ and ending at t. This job is at worst of size C. The two black arrows illustrate the position of t_1^i with respect to that of t_2^i.

▶ **Case 1:** We consider the case where $t_2 - t < 1$. We are faced with two subcases:

- Either $t_1 = et - (e-1)t_2 > t-1$. In that subcase, no job can arrive after t_1 with a deadline smaller than t_2. Therefore $u(t, t_1, t_2) = 0$, and $\pi^{(\mathrm{BKP})}(t) = 0$. This subcase is illustrated in Fig. 4 by the tuple (t, t_1^a, t_2^a).

- Or $t_1 = et - (e-1)t_2 \leq t-1$. Here, potentially, one job can arrive at $t-1$ and end at t. We introduce the variable $\beta \in \mathbb{R}$ such that $t_2 = t + \beta$ and $t_1 \leq t-1$. This limit case (the earliest t_2, under these conditions, such that one job can be taken into account in the (BKP) speed computation) leads to:

$$\begin{aligned} & t_1 = t-1 \\ \iff \quad & et - (e-1)(t+\beta) = t-1 \\ \iff \quad & \beta = \frac{1}{e-1}. \end{aligned}$$

  Therefore the maximal value for $u(t, t_1, t_2)$ is $\frac{C}{\beta}$ and is reached for $t_2 = t + \beta$. Note that $\frac{C}{\beta}$ is independent of t. This subcase is illustrated in Fig. 4 by the tuple (t, t_1^b, t_2^b).

▶ **Case 2:** We consider the case where $t_2 \geq t+1$. In this case the contributing work is bounded by:

$$u(t, t_1, t_2) \leq C\lfloor t - t_1 + 1 \rfloor. \tag{51}$$

because, even when $t = t_1$, one job can arrive at t and be taken into account by (BKP) (hence the "+1"). It follows that the speed computed by (BKP) is:

$$\begin{aligned} \pi^{(\mathrm{BKP})}(t) &\leq C \max_{t_2 > t} \left\{ \frac{\lfloor t - et + (e-1)t_2 + 1 \rfloor}{t_2 - t} \right\} \\ &\leq C \max_{t_2 > t} \left\{ \frac{\lfloor (e-1)(t_2 - t) + 1 \rfloor}{t_2 - t} \right\}. \end{aligned} \tag{52}$$

To reason about Eq. (52), we introduce the variable $\gamma \in \mathbb{R}_+$, such that $t_2 = t + 1 + \gamma$. Accordingly, $\pi^{(\mathrm{BKP})}$ depends only on γ:

$$\pi^{(\mathrm{BKP})}(\gamma) \leq C \max_{\gamma \in \mathbb{R}_+} \left\{ \frac{\lfloor (e-1)(1+\gamma) + 1 \rfloor}{1+\gamma} \right\}. \tag{53}$$

Because of the floor operator, $(e-1)(1+\gamma)+1$ must be in $\mathbb{N}$ for the fraction $\frac{\lfloor (e-1)(1+\gamma)+1 \rfloor}{1+\gamma}$ to be maximized, and since $e-1$ is irrational, there must exist $k \in \mathbb{N}$ such that:

$$1 + \gamma = \frac{k}{e-1}. \tag{54}$$

It follows that:

$$\frac{\lfloor (e-1)(1+\gamma) + 1 \rfloor}{1+\gamma} = \frac{(e-1)(1+\gamma)+1}{1+\gamma} = e - 1 + \frac{1}{1+\gamma}. \tag{55}$$

Now, since γ is positive, we have $1 + \gamma \geq 1$, so:

$$\frac{k}{e-1} \geq 1 \iff k \geq e - 1. \tag{56}$$

The function $\gamma \mapsto e - 1 + \frac{1}{1+\gamma}$ is decreasing and γ has to satisfy the condition of the Inequality (56). Therefore the maximum of Eq. (53) is reached for the smallest $k \in \mathbb{N}$ (*i.e.*, the smallest possible γ) such that Inequality (56) is satisfied:

$$k = \min_{j \in \mathbb{N}} \{ j \geq e - 1 \} = \lceil e - 1 \rceil. \tag{57}$$

Finally, by replacing k by its value in Eq. (54), we obtain:

$$\gamma = \frac{\lceil e-1 \rceil}{e-1} - 1 = \frac{3-e}{e-1} \simeq 0.16. \tag{58}$$

From Eqs. (53), (55), and (58), it follows that:

$$\pi^{(\mathrm{BKP})}(t) \leq C \left(e - 1 + \frac{e-1}{2} \right) = \frac{3}{2}(e-1)C \simeq 2.577\, C. \tag{59}$$

Putting Case 1 and Case 2 together, we obtain:

$$\begin{aligned}
\forall t, \pi^{(\mathrm{BKP})}(t) &= \max\left\{ \max_{t_2 \geq t+1} \left\{ \frac{u(t, et-(e-1)t_2, t_2)}{t_2 - t} \right\}, \right. \\
&\qquad \left. \max_{t+1>t_2>t} \left\{ \frac{u(t, et-(e-1)t_2, t_2)}{t_2 - t} \right\} \right\} \\
&\leq \max\left\{ \max_{\gamma \in \mathbb{R}_+} \left\{ \left((e-1) + \frac{1}{1+\gamma} \right) C \right\}, (e-1)C \right\} \\
&\leq \max_{\gamma \in \mathbb{R}_+} \left\{ \left((e-1) + \frac{1}{1+\gamma} \right) C \right\} \\
&\leq \frac{3}{2}(e-1)C \simeq 2.577\, C. \qquad (60)
\end{aligned}$$

We now want to establish a *lower bound* on the maximal speed of (BKP), by using Eq. (60). If we are in the particular case depicted in Figure 2 where $t = n + \Delta - 1$ and $t_2 = t + 1 + \gamma$, then we have $t_1 = et - (e-1)t_2 \in \mathbb{N}$ by definition of t_1 in Def. 10. Under these conditions and according to the previous computations done for the upper bound case, we have for this particular t:

$$\pi^{(\mathrm{BKP})}(t) = \frac{3}{2}(e-1)C. \qquad (61)$$

Since the lower bound of Eq. (61) is equal to the upper bound of Eq. (60), we can conclude that:

$$\text{(BKP) is feasible} \iff S_{\max} \geq \frac{3}{2}(e-1)C.$$

□

7.3 Feasibility analysis of (BKP) with $\mathcal{T} = \mathbb{R}$

When the speed decision times are real numbers, another feasibility condition holds for (BKP) policy, stated in Theorem 4.

Theorem 4. (BKP) *is feasible with* $\mathcal{T} = \mathbb{R} \iff S_{\max} \geq eC$.

Proof. Let us consider the same three following variables t, t_1, and t_2, as in the proof of Theorem 3. The only difference is the fact that t is in $\mathbb{R}$ instead of $\mathbb{N}$.

Similarly to the proof of Theorem 3, we have to prove the following equivalence:

$$\text{(BKP) is feasible with } \mathcal{T} = \mathbb{R} \iff \forall t \in \mathbb{R},\ S_{\max} \geq \pi^{(\mathrm{BKP})}(t). \qquad (62)$$

To do so, we will use the same method as in the previous proof, *i.e.*, we determine an upper and a lower bound for the maximal speed of (BKP). To begin, we will find an *upper bound* on the maximal speed of (BKP). We introduce the variable $\beta \in \mathbb{R}$ such that $t_2 = t + \beta$. The set of jobs that are taken in consideration in (BKP) speed computation belongs to an interval of length $e\beta$, because:

$$t_2 - t_1 = t + \beta - et_1 + (e-1)(t+\beta) = e\beta.$$

This situation is depicted in Figure 5.

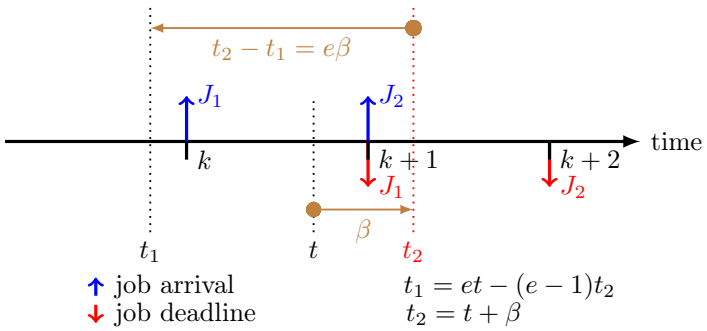


Figure 5: Real speed decision times, *i.e.*, $t \in \mathbb{R}$ and $1 \leq t_2 - t_1 < 2$. Two jobs $J_1 = (k, C, 1)$ and $J_2 = (k+1, C, 1)$ are represented.

Let $n = \lfloor t_2 - t_1 \rfloor$, hence $n \leq t_2 - t_1 < n + 1$. Then at most $n + 1$ jobs can arrive in the $[t_1, t_2)$ interval and at most n of them can have a deadline before t_2, therefore $u(t, t_1, t_2) \leq nC$ so $\pi^{(\mathrm{BKP})}(t) \leq \frac{nC}{n/e} = eC$. For all t in $\mathbb{R}$, an upper bound on (BKP) maximal speed is thus:

$$\pi^{(\mathrm{BKP})}_{\max} \leq eC. \tag{63}$$

Now we consider the particular situation, where $\beta = 1/e$, $t_2 = 1$ and there is one job of size C with deadline 1 that arrives at time 0. In that case, $t_1 = 0$, $t = 1 - \frac{1}{e}$, and $t_2 = 1$. It follows that $u(t, t_1, t_2) = C$ and:

$$\pi^{(\mathrm{BKP})}(t) = \frac{C}{\beta} = eC. \tag{64}$$

As a conclusion, since the upper bound of Eq. (63) is reached (see Eq. (64)), we have:

$$\text{(BKP) is feasible} \iff eC \leq S_{\max}.$$

□

8 Feasibility of the Markov Decision Process speed policy (MP)

This last policy shows that one can get the best of both worlds: An energy optimal policy whose feasibility region is maximal, at the price of statistical information about future jobs.

8.1 Definition of (MP) [3]

In this section we assume that the job sequence $\{J_i\}_{i \in \mathbb{N}}$ is endowed with a probability distribution on (r_i, c_i, D_i). The precise values of the probabilities that a job is released at time r_i, is of size c_i, or has a relative deadline D_i are indeed important to compute the speed used at any time t by the on-line speed policy (MP), but they will not play a role in the feasibility analysis on (MP), as seen in the following.

To define (MP), we first introduce the *state* of the system at time t that gathers all the information useful to decide which speed to use at time t. Since all job features are integer numbers and the relative deadline is smaller than Δ, the current information at time t can be summarized in the vector $(w_t(t+1), w_t(t+2), \ldots, w_t(t+\Delta))$, which will be called the state at time t in the following, and denoted x_t.

Under this framework, we define the transition matrix $P_s(x, x')$, that gathers the probabilities to go from state x to state x' in one time step when the processor speed is s. The construction of this transition matrix requires to know the distribution of the release times, the sizes and the deadlines of future jobs. This knowledge may come from statistical analysis of the jobs in a training phase preceding the deployment of the speed policy in the system, or can even be learned on-line: the system adjusts its estimation of the optimal speed at each step using a no-regret algorithm (see for example [9]).

For any on-line policy π, the long run average *expected* energy consumption per time-unit for policy π under the probability transition P_s, noted Q_π, is defined as:

$$Q_\pi(x_0) = \mathbb{E}\left(\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathit{Energy}(\pi(t))\right). \tag{65}$$

where x_0 is the initial state of the process, and $\mathit{Energy}(s)$ is the energy consumption of the processor when the speed is s during one unit of time.

An optimal speed policy π^* minimizes the average expected energy consumption per time-unit given in Eq. (65). Therefore, the speed policy (MP) is defined as:

Definition 11 ((MP) policy)**.** *At each time $t \in \mathcal{T}$, the job that has the earliest deadline is executed at speed:*

$$\pi^{(\mathrm{MP})}(t) \text{ is such that } Q_{\pi^{(\mathrm{MP})}}(x_0) = \inf_{\{\pi \,|\, \forall t\in\mathcal{T},\ \pi(t)\geq w_t^\pi(t+1)\}} Q_\pi(x_0). \tag{66}$$

Remark 4. *Several remarks are in order:*

- *The optimal policy minimizing the expected energy consumption may not be unique. In the following we consider one arbitrary such speed policy. This does not matter because feasibility as well as the expected energy consumption is the same for all of them.*
- *This definition of $\pi^{(\mathrm{MP})}$ is not constructive but when the set of speeds is finite, then $\pi^{(\mathrm{MP})}$ can be constructed explicitly using for example the Policy Iteration algorithm (see for instance [10]).*
- *It can also be shown that an optimal policy,* i.e.*, a solution of Eq.* (66)*, is independent of x_0. This is outside the scope of this paper.*

8.2 Feasibility analysis of (MP)

Theorem 5 gives the value of $S_{\max}$ that ensures feasibility:

Theorem 5. (MP) *is feasible* $\iff S_{\max} \geq C$.

Proof. We distinguish the cases where the speed decision times are integer and real numbers.

♦ **The speed decision times are integer numbers: $\mathcal{T} = \mathbb{N}$.**
By definition, (MP) completes all the jobs before their deadline by construction: $\pi^{(\mathrm{MP})}(t) \geq w_t^{(\mathrm{MP})}(t+1)$. Therefore, (MP) is feasible if at any time $t \in \mathbb{N}$, $\pi^{(\mathrm{MP})}(t) \leq S_{\max}$.

1. Case $S_{\max} < C$: In that case, no speed policy can guarantee feasibility as shown in Proposition 2.

2. Case $S_{\max} \geq C$: To prove the result, we first modify the *Energy* function as follows: For all speeds $s > S_{\max}$, we set $\mathit{Energy}(s) = \infty$. For $s \leq S_{\max}$, the *Energy* function remains unchanged. This modification is valid because the processor cannot use speeds larger than $S_{\max}$ anyway. Therefore, the energy consumption for such unattainable speeds can be arbitrarily set to any value. The benefit of using this modification is the following. Instead of constraining the speed to remain smaller than $S_{\max}$, we let the scheduler use unbounded speeds, but this incurs an infinite consumption. A test to check if a policy uses speeds larger than $S_{\max}$ is that its average energy consumption will be infinite.

Starting from an empty system with no pending job, *i.e.*, $x_0 = (0, 0, \ldots, 0)$, we define the following naive policy $\tilde{\pi}$:

$$\forall t \in \mathbb{N},\ \tilde{\pi}(t) := c_t \quad \text{where } c_t = \sum_{J_i=(r_i,c_i,d_i)} \{c_i | r_i = t\}. \tag{67}$$

In other words, c_t is the amount of work that arrived at time t, which is by definition less than C. The policy $\tilde{\pi}$ is feasible because it never uses a speed larger than $C \leq S_{\max}$ and all work is executed as fast as possible (within one time slot after its arrival). Furthermore, since for any t, $\tilde{\pi}(t) \leq C$, its long run expected energy consumption per time unit satisfies $Q_{\tilde{\pi}}(x_0) \leq \mathit{Energy}(C)$.

The optimal policy, being optimal in energy, satisfies $Q_{\pi^{(\mathrm{MP})}}(x_0) \leq Q_{\tilde{\pi}}(x_0)$, hence $Q_{\pi^{(\mathrm{MP})}}(x_0) \leq \mathit{Energy}(C)$. Therefore, (MP) is feasible by construction and never uses a speed larger than $S_{\max}$.

♦ **The speed decision times are real numbers: $\mathcal{T} = \mathbb{R}$.**
When the speed can be changed at any time $t \in \mathbb{R}$, the average expected energy consumption of a policy π becomes

$$Q_\pi(x_0) = \lim_{T\to\infty} \mathbb{E}\left(\frac{1}{T}\int_0^T \mathit{Energy}(\pi(t))dt\right). \tag{68}$$

When $S_{\max} < C$, then Proposition 2 says that no policy can be feasible, so neither is (MP).

Now, let us consider that $S_{\max} \geq C$. The optimal policy $\pi^{(\mathrm{MP})}$ is defined by taking the inf in Eq. (66), not over the set $\mathcal{A}^{\mathbb{N}} = \{\pi \,|\, \forall t \in \mathbb{N},\ \pi(t) \geq w_t^\pi(t+1)\}$ anymore, but over the set $\mathcal{A}^{\mathbb{R}} = \{\pi \,|\, \forall t \in \mathbb{R},\ w_t^\pi(t) \leq 0\}$. Since $\mathcal{A}^{\mathbb{N}} \subset \mathcal{A}^{\mathbb{R}}$ (see Prop. 3), it follows that $Q^{\mathcal{A}^{\mathbb{R}}}_{\pi^{(\mathrm{MP})}} \leq Q^{\mathcal{A}^{\mathbb{N}}}_{\pi^{(\mathrm{MP})}}$.

We have proven above that if $S_{\max} \geq C$, then $Q^{\mathcal{A}^{\mathbb{N}}}_{\pi^{(\mathrm{MP})}}$ is finite (it is less than $\mathit{Energy}(C)$). Therefore, $Q^{\mathcal{A}^{\mathbb{R}}}_{\pi^{(\mathrm{MP})}} \leq \mathit{Energy}(C)$. This implies that the optimal policy never uses speeds larger than $S_{\max}$, as in the discrete case. In conclusion, the (MP) policy with $\mathcal{T} = \mathbb{R}$ is feasible if and only if $S_{\max} \geq C$. □ □

9 Summary and Comparison of the four Policies

Table 2 summarizes the necessary and sufficient feasibility conditions on $S_{\max}$ for the four on-line speed policies (OA), (AVR), (BKP), and (MP), both in the integer and real speed decision times cases.

On-line speed policy	Necessary and sufficient feasibility condition	
Speed decision times	$\mathbb{N}$	$\mathbb{R}$
(OA)	$S_{\max} \geq C(h_{\Delta-1}+1)$	
(AVR)	$S_{\max} \geq Ch_{\Delta}$	
(BKP)	$S_{\max} \geq \frac{3}{2}(e-1)C$ $(\simeq 2.577\,C)$	$S_{\max} \geq eC$ $(\simeq 2.718\,C)$
(MP)	$S_{\max} \geq C$	

Table 2: Necessary and sufficient feasibility condition of the four on-line speed policies.

For a given on-line speed policy π, we define the *feasibility region* $\mathcal{F}_\pi$ as the set of all triples $(C, \Delta, S_{\max})$ such that π is feasible. We rely on this notion of feasibility region to compare the policies. We make the following remarks:

1. By observing the (AVR) and (OA) feasibility bounds, we can remark that their maximal speeds are asymptotically identical when Δ becomes large. However, since for all $\Delta \in \mathbb{N}$ we have $\frac{1}{\Delta} \leq 1$, (AVR) and (OA) satisfy the following equation:

$$\pi_{\max}^{(\mathrm{AVR})} \leq \pi_{\max}^{(\mathrm{OA})}.$$

   Consequently, since the maximal speed reached by (OA) is faster than the maximal speed reached by (AVR), (AVR) has a better feasibility than (OA), in the sense that the feasibility region of (AVR) includes the feasibility region of (OA):

$$\mathcal{F}_{(\mathrm{OA})} \subset \mathcal{F}_{(\mathrm{AVR})}.$$

2. Let us now compare the feasibility regions of (AVR) and (OA) with that of (BKP). Since this comparison depends on the harmonic number h_Δ, we display in Table 3 the approximated values of h_Δ and $h_{\Delta-1}+1$ (rounded down) for different values of Δ:

Δ	2	3	4	5	6	7	8	9
h_Δ	1.500	1.833	2.083	2.283	2.450	2.593	2.717	2.828
$h_{\Delta-1}+1$	2.000	2.500	2.833	3.083	3.283	3.450	3.593	3.717

Table 3: Values of the harmonic numbers h_Δ and of $h_{\Delta-1}+1$ (with 3 significant digits).

   Since the feasibility bounds of (BKP) are $\frac{3}{2}(e-1)C \simeq 2.577\,C$ when $\mathcal{T} = \mathbb{N}$ and $eC \simeq 2.718\,C$ when $\mathcal{T} = \mathbb{R}$, we compare in Table 4 the feasibility regions of (AVR), (OA), and (BKP) depending on the value of Δ:

Feasibility regions $\mathcal{F}_\pi$	$\mathcal{F}_{(\mathrm{AVR})} \subset \mathcal{F}_{(\mathrm{BKP})}$	$\mathcal{F}_{(\mathrm{OA})} \subset \mathcal{F}_{(\mathrm{BKP})}$	$\mathcal{F}_{(\mathrm{OA})} \subset \mathcal{F}_{(\mathrm{AVR})}$
Integer decision times	$\forall\Delta \geq 7$	$\forall\Delta \geq 4$	$\forall\Delta \in \mathbb{N}$
Real decision times	$\forall\Delta \geq 9$	$\forall\Delta \geq 4$	$\forall\Delta \in \mathbb{N}$

Table 4: Feasibility region comparisons for (OA), (AVR), and (BKP).

   (MP) is not present in Table 4 because it is clear from Table 2 that (MP) has the largest feasibility region:

$$\forall\pi \in \{(\mathrm{OA}), (\mathrm{AVR}), (\mathrm{BKP})\}, \quad \mathcal{F}_\pi \subset \mathcal{F}_{(\mathrm{MP})}$$

3. Unlike (OA) and (AVR), the (BKP) feasibility bounds are independent of the maximal deadline Δ. This means that the (BKP) feasibility regions do not change when Δ grows, whereas for (OA) and (AVR) the feasibility region decreases to the empty set when Δ increases.

4. For (BKP), one can wonder whether the parameter e can be changed in Eq. (49) to improve its feasibility (see Theorems 3 and 4). If we replace e by a parameter α in the definition of (BKP), we obtain a variant policy denoted (BKP_α). The feasibility region becomes $\mathcal{F}_{(\text{BKP}_\alpha)} = \{S_{\max} \geq \alpha C\}$ for any $\alpha \geq e$, by using the same proof as in Section 7. However, if $\alpha < e$, then it can be shown that (BKP_α) is not feasible even with $S_{\max} = +\infty$. It follows that $\alpha = e$ is the best possible choice.

5. Finally, (MP) is optimal both in terms of energy and feasibility, so it is a good candidate to be used on-line to process real-time jobs. Its drawback, however is twofold: on the one hand its complexity, the time and space complexity to compute $\pi^{(\text{MP})}(t)$ being $O(C^\Delta)$ (see [3]); and on the other hand the requirement to know the probability distributions on r_i, c_i, and D_i.

10 Conclusion

Adjusting the processor speed dynamically in hard real-time systems allows the energy consumption to be minimized. This is achieved by an *on-line speed policy*, the goal of which is to determine the speed of the processor to execute the current, not yet finished, jobs. Several such policies have been proposed in the literature, including (OA), (AVR), (BKP), and (MP). Since they are targeting hard real-time systems, they must satisfy two constraints: each real-time job must finish before its deadline, and the maximal speed used by the policy must be less than or equal to the maximal speed $S_{\max}$ available on the processor. We call the conjunction of these two constraints the *feasibility* condition of the policy.

In this paper, we have established for each of the four policies (OA), (AVR), (BKP), and (MP), a necessary and sufficient condition for the feasibility. (OA) is feasible if and only if $S_{\max} \geq C(h_{\Delta-1}+1)$. (AVR) is feasible if and only if $S_{\max} \geq Ch_\Delta$. (BKP) is feasible if and only if $S_{\max} \geq eC$ when the processor speed can change at any time, and $S_{\max} \geq \frac{3}{2}(e-1)C$ when the processor speed can change only upon the arrival of a new job (for the other policies, the times at which the processor speed can change has no impact on the feasibility condition). Finally, (MP) is feasible if and only if $S_{\max} \geq C$. This is optimal because, as shown in Proposition 2, the necessary condition of feasibility of all on-line policies is $S_{\max} \geq C$. Therefore, (MP) is optimal in terms of feasibility in addition to being optimal in energy (on average), but it requires the statistical knowledge of the arrival times, execution times, and deadlines of the jobs, and it is more expensive to compute than the other speed policies.

Appendix

A Uniqueness of the solution of Eq. (38)

Let us rewrite Eq. (38) $\forall t \in [k, k+1[$ as follow:

$$\pi(t) = \max_{v>t} \frac{w_k(v) - \int_k^t \pi(x)\mathrm{d}x}{v-t}. \tag{69}$$

The goal of this part is to prove that there exists a unique solution π for this equation. By doing an appropriate variable shift on Eq. (69), $u = v - t$, we obtain:

$$\pi(t) = \sup_{u>0} \frac{w_k(u+t) - \int_k^t \pi(x)\mathrm{d}x}{u}. \tag{70}$$

By defining $W(t) = \int_k^t \pi(x)\mathrm{d}x$, and integrating Eq. (70) between k and t, we obtain:

$$W(t) = \int_k^t \sup_{u>0} \frac{w_k(u+s) - W(s)}{u}\mathrm{d}s. \tag{71}$$

Let us now define the function $F(s,x)$ as follow:

$$F(s,x) = \sup_{u>0} \left(\frac{f_s(u) - x}{u} \right) \tag{72}$$

where $f_s(\cdot) = w_k(\cdot + s)$. Then $f_s(\cdot)$ is such that:

1. f_s is an increasing function bounded by $C\Delta \in \mathbb{R}^+$.
2. $f_s(0) = w_k(s) = 0$ because $w_k(k) = 0$ by feasibility and no job arrives between k and s.
3. f_s satisfies:
$$\lim_{\substack{t\to 0 \\ t\geq 0}} \frac{f_s(t)}{t} = 0 \tag{73}$$
because $w_k(s)$ is constant for $s \in [k, k+1)$.

The function $W(t) = \int_k^t \pi(u)\mathrm{d}u$ also satisfies the following integro-differential equation:

$$W(t) = \int_k^t F(s, W(s))\mathrm{d}s. \tag{74}$$

Lemma 2. *There exists a unique solution W to Eq. (74).*

Proof. First, let us show in Lemmas (3)-(4) that the function $F(s,x)$ is Lipschitz in x.

Lemma 3. *Let $t_0 > 0$ be the first time such that the sup of $F(s,0)$ is reached. Then $F(s,x)$ is a $\frac{1}{t_0}$-Lipschitz function in x.*

Proof. For the proof of Lemma 3, we will note $g(x) = F(s, x)$. Let $x, y \in [0, a]$ (where a is an arbitrary positive number). We want to prove that:

$$\exists k \in \mathbb{R},\ |g(x) - g(y)| \leq k|x - y|. \tag{75}$$

Let us compute the difference $|g(x) - g(y)|$:

$$|g(x) - g(y)| \quad = \quad \left| \sup_{t>0} \frac{f_s(t) - x}{t} - \sup_{t>0} \frac{f_s(t) - y}{t} \right| \tag{76}$$

Since, by assumption, the function $f_s(t)$ is bounded by a, the sup for $g(x)$ is reached for a certain value of t, noted t_x.

By definition of the sup, we have $\sup_{t>0} \frac{f_s(t)-y}{t} \geq \frac{f_s(t_x)-y}{t_x}$, hence Eq. (76) becomes:

$$\begin{aligned} |g(x) - g(y)| &\leq \left| \frac{f_s(t_x) - x}{t_x} - \frac{f_s(t_x) - y}{t_x} \right| \\ |g(x) - g(y)| &\leq \left| \frac{y - x}{t_x} \right| \end{aligned} \tag{77}$$

Now let us prove Lemma (4), which states that t_x is an increasing function in x.

Lemma 4. *Let t_x be the function of x such that:*

$$\forall a \in \mathbb{R},\ \forall x \in [0, a],\ t_x : x \longmapsto \operatorname*{argmax}_t \frac{f_s(t) - x}{t}.$$

Then t_x is an increasing function of x.

Proof. Let $x, y \in [0, a]$ such that $x \leq y$, and let t_y be such that:

$$\frac{f_s(t_y) - y}{t_y} = \max_t \left(\frac{f_s(t) - y}{t} \right) \tag{78}$$

The goal is to prove Eq. (79) below:

$$\frac{f_s(t) - x}{t} \leq \frac{f_s(t_y) - x}{t_y}. \tag{79}$$

By definition of the max, we have for any defined function f_s:

$$\forall t \in \mathbb{R},\ f_s(t) \leq \frac{f_s(t_y) - y}{t_y} t + y. \tag{80}$$

We now define two lines:

- the line L_1 that corresponds to the slope for the maximal value of y, *i.e.*, the line that links the points $(0, y)$ and $(t_y, f_s(t_y))$; its equation corresponds to the left part of Eq. (80);
- and the line L_2 that links $(0, x)$ on the ordinate axis and the point $(t, f_s(t))$.

The functions $t \longmapsto L_1(t)$ and $t \longmapsto L_2(t)$ correspond to all the points of their respective lines.

By definition of $L_1(t)$, we have $f_s(t) \leq L_1(t)$. Moreover as time $t \geq t_y$, by construction of line L_2, we have $L_2(t) \leq L_1(t)$. Since $x \leq y$, we also have $L_2(0) \leq L_1(0)$. All these inequalities on some points of the two lines L_1 and L_2 imply that $L_1(t_y) \geq L_2(t_y)$. The expressions of the functions $L_1(t)$ and $L_2(t)$ lead to the following inequality:

$$f_s(t_y) \geq \frac{f_s(t) - x}{t} t_y + x \iff \frac{f_s(t_y) - x}{t_y} \geq \frac{f_s(t) - x}{t}.$$

Eq. (79) is therefore satisfied, and so the function $x \longmapsto t_x$ is an increasing function. □ □

Using Eq. (73), the fact that f_s is an increasing function, and the fact that $f_s(0) = 0$, the first time t such that $F(t, 0) > 0$ is strictly larger than 0, and as we want to determine for all t the sup of $F(t, 0)$, then t_0 is strictly positive.

Since t_x is an increasing function of x by Lemma (4), and since $t_0 > 0$, then Eq. (77) becomes:

$$|g(x) - g(y)| \quad \leq \quad \frac{1}{t_0}|y - x|. \tag{81}$$

Eq. (81) concludes that g is $\frac{1}{t_0}$-Lipschitz. □ □

Since $F(s, x)$ is Lipschitz in x, the Picard-Lindelof theorem allows us to concludes that there exists a unique solution $W(t)$ for the Eq. (74).

Therefore Eq. (74) can be rewritten as follow:

$$\pi(t) \quad = \quad \sup_{u>0} \frac{f_s(u) - W(t)}{u}. \tag{82}$$

By Eq. (82), π is a function of W, so π is also unique. □ □

B Concavity of the executed work by (OA) for a given w

In this appendix we provide a more exhaustive study of the speed policy (OA). We show that the work executed by (OA) is the convex envelope of the graph of the remaining work function $w(.)$, when $w(.)$ is fixed (*i.e.*, all the jobs arrive at time 0). Using the same notation as in the previous appendix (the index $k = 0$ is dropped in w_0), we define $W(t) = \int_0^t \pi^{(\mathrm{OA})}(u)\mathrm{d}u$, the amount of work executed by (OA) from time 0 to time t. It is such that:

$$W(t) = \int_0^t \sup_{u \geq 0} \frac{w(u+s) - W(s)}{u} \mathrm{d}s = \int_0^t \sup_{v \geq s} \frac{w(v) - W(s)}{v - s} \mathrm{d}s. \tag{83}$$

$W(t)$ corresponds to the quantity of work executed between 0 and t, and the goal of this part is to show that $W(t)$ is the smallest concave function that is above $w(t)$.

Lemma 5. *Let w be any real non-decreasing function that admits right-derivatives everywhere (not necessarily staircase), with $w(0) = 0$. Then $W(t)$ as defined in Eq.* (83) *satisfies the following properties:*

1. *W is continuous, $W(0) = 0$, and $\forall t \geq 0$, $W(t) \geq w(t)$.*
2. *W is non-decreasing in w.*
3. *If w is concave, then $W(t) = w(t)$.*
4. *W is concave.*
5. *$W = \widehat{w}$ where $\widehat{w}$ is the convex hull of w.*

Proof.

1. $W(t)$ being an integral from 0 to t, W is continuous, $W(0) = 0$, and W has right-derivatives everywhere: $W'_+(t) = \sup_{u \geq 0} \frac{w(u+t)-W(t)}{u}$. Let us denote by $w'_+(\cdot)$ the right-derivative of w: $w'_+(t) = \lim_{u \to 0, u \geq 0}(w(t+u) - w(t))/u$. Then $w'_+(t) \leq \sup_{u \geq 0} \frac{w(u+t)-w(t)}{u}$. Since $w(0) = 0 = W(0)$, then by Petrovitsch Theorem on differential inequalities, [11], we have $W(t) \geq w(t)$ for all $t \geq 0$.

2. By definition of the function W, it is a non-decreasing function in w.

3. Let us suppose that w is concave. By replacing, in the right part of Eq. (83), W by w, one gets inside the integral:

$$\sup_{v \geq s} \frac{w(v) - w(s)}{v - s}. \tag{84}$$

Since w is concave by assumption, it is right and left differentiable at any point t. This means that w'_+, the right-derivative of w, is decreasing. Therefore the sup is reached when v goes to s. We thus have:

$$\sup_{v \geq s} \frac{w(v) - w(s)}{v - s} = w'_+(s). \tag{85}$$

By using Eq. (85) and replacing in Eq. (83), we obtain:

$$\int_0^t \sup_{v \geq s} \frac{w(v) - w(s)}{v - s} \mathrm{d}s = \int_0^t w'_+(u) \mathrm{d}u = w(t). \tag{86}$$

The last equality in Eq. (86) is due to the fact that w is concave. Indeed, since w is concave, its derivative is defined on the whole interval $[k, t]$ except for a finite number of values u. The integral does not depend on these points, so we have Eq (86).

To conclude, w is a solution of Eq (83) and so $W = w$ by uniqueness of the solution.

4. For any $t \geq 0$, let L_t be the right tangent of W at the point t. The equation of the line L_t is: $L_t(v) = W(t) + W'_+(t)(v-t)$. Since $W(t) \geq w(t)$, we have for all $v \geq t$, $L_t(v) = W(t) + W'_+(t)(v-t) \geq w(v)$.

   If we replace w by L_t in the definition of W, we get a new function W_L that is larger than W by item 2 and is equal to L_t by item 3. This means that W is below its right tangents.

   Now, this implies that W is concave: By contradiction, let $x < y$ be such that $\forall z \in [x, y]$, we have $W(z) < A(z)$, where $A(\cdot)$ is the affine interpolation between $W(x)$ and $W(z)$. Since W is below its right tangents, then $W(y) \leq W(z) + W'_+(z)(y-z)$. This implies that $W'_+(z)$ is larger than the slope of $A(\cdot)$: in other words, $W'_+(z) \geq \frac{W(y)-W(x)}{y-x}$. By integrating this inequality from x to z, we get $W(z) \geq A(z)$. This contradicts the initial assumption that $W(z) < A(z)$.

   The final conclusion is that W is always above its affine interpolation, hence W is concave.

5. Let us prove first that $W \geq \widehat{w}$. By definition of the convex hull, we know that $w \leq \widehat{w}$, and as W is an increasing function in w, $W \leq W_{\widehat{w}}$, where $W_{\widehat{w}}$ is the function W where w is replaced by $\widehat{w}$. Since $\widehat{w}$ is concave, we then have by item 3 the fact that $\widehat{w} = W_{\widehat{w}}$. This implies:

   $$W \leq \widehat{w}.$$

   Now we prove the other inequality, *i.e.*, that $\widehat{w} \leq W$. By item 1, we get $W \geq w$. By item 4, we know that W is concave. Since $\widehat{w}$ is the smallest concave function above w, we finally have:

   $$W \geq \widehat{w}.$$

   We therefore conclude that:

   $$W = \widehat{w}.$$

□ □

References

[1] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in *IEEE Annual Foundations of Computer Science*, pp. 374–382, 1995.

[2] N. Bansal, T. Kimbrel, and K. Pruhs, "Speed scaling to manage energy and temperature," *J. of the ACM*, vol. 54, no. 1, 2007.

[3] B. Gaujal, A. Girault, and S. Plassart, "Dynamic speed scaling minimizing expected energy consumption for real-time tasks," Tech. Rep. hal-01615835, Inria, 2017.

[4] J. Chen, N. Stoimenov, and L. Thiele, "Feasibility analysis of on-line DVS algorithms for scheduling arbitrary event streams," in *Real-Time Systems Symposium, RTSS'09*, (Washington (DC), USA), pp. 261–270, IEEE, Dec. 2009.

[5] L. Thiele, S. Chakraborty, and M. Naedele, "Real-time calculus for scheduling hard real-time systems," in *International Symposium on Circuits and Systems, ISCAS'00*, pp. 101–104, IEEE, May 2000.

[6] R. Jejurikar and R. Gupta, "Procrastination scheduling in fixed priority real-time systems," in *Conference on Languages, Compilers, and Tools for Embedded Systems, LCTES'04*, (Washington (DC), USA), pp. 57–66, ACM, June 2004.

[7] J. Augustine, S. Irani, and C. Swamy, "Optimal power-down strategies," in *Symposium on Foundations of Computer Science, FOCS'04*, (Rome, Italy), pp. 530–539, IEEE, Oct. 2004.

[8] C. Liu and J. Layland, "Scheduling algorithms for multiprogramming in hard real-time environnement," *J. of the ACM*, vol. 20, pp. 46–61, Jan. 1973.

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, second ed., 2018.

[10] M. L. Puterman, *Markov Decision Process : Discrete Stochastic Dynamic Programming*. Wiley, wiley series in probability and statistics ed., Feb. 2005.

[11] M. Petrovitsch, "Sur une manière d'étendre le théorème de la moyence aux équations différentielles du premier ordre," *Math. Ann.*, vol. 54, no. 3, pp. 417–436, 1901.

[12] F. Baccelli and J. Mairesse, "Ergodic Theorems for Stochastic Operators and Discrete Event Networks," in *Idempotency* (C. U. Press, ed.), Publications of the Newton Institute, pp. 171–208, Cambridge University Press, 1998.

[13] A. Marshall and I. Olkin, *Inequalitites: Theory of Majorization and Its Applications*, vol. 143 of *Mathematics in Science and Engineering*. Academic Press, 1979.

[14] D. Bertsekas and J. Tsitsiklis, *Neuro-dynamic programming*. Belmont, Mass., USA: Athena Scientific, 1996.

[15] A. Müller and D. Stoyan, *Comparison Methods for Stochastic Models and Risks*. No. ISBN: 978-0-471-49446-1 in Wiley Series in Probability and Statistics, Wiley, 2002.

[16] H. Yun and J. Kim, "On energy-optimal voltage scheduling for fixed priority hard real-time systems," *ACM Trans. Embed. Comput. Syst.*, vol. 2, no. 3, pp. 393–430, 2003.

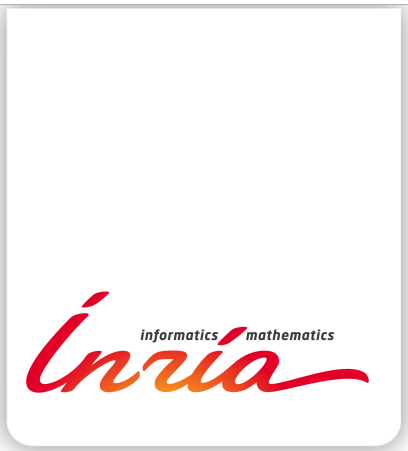
[17] M. Li and F. F. Yao, "An efficient algorithm for computing optimal discrete voltage schedules," *SIAM J. Comput.*, vol. 35, pp. 658–671, 2005.

[18] B. Gaujal, N. Navet, and C. Walsh, "Shortest path algorithms for real-time scheduling of fifo tasks with minimal energy use," *ACM Trans. on Emb. Comput. Syst.*, vol. 4, no. 4, 2005.

[19] J. Lorch and A. Smith, "Improving dynamic voltage scaling algorithms with PACE," in *ACM SIGMETRICS 2001 Conference*, pp. 50–61, 2001.

[20] F. Gruian, "On energy reduction in hard real-time systems containing tasks with stochastic execution times," in *IEEE Workshop on Power Management for Real-Time and Embedded Systems*, pp. 11–16, 2001.

[21] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," *SIGOPS Oper. Syst. Rev.*, vol. 35, pp. 89–102, Oct. 2001.



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399